НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.491 : 004.492

ПОГОДЖЕНО
Декан факультету
Інформаційних технологій
_____/ Глазунова О.Г., д.п.н, проф. /
підпис ПІБ, вчене звання і ступінь
«___»_____2024 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
Комп'ютерних систем, мереж та кібербезпеки
_____/ Касаткін Д.Ю., к.п.н., доцент. /
підпис ПІБ, вчене звання і ступінь
«___»_____2024 р.

МАГІСТЕРСЬКА РОБОТА

На тему: «Дослідження методів оцінювання ризиків інформаційної безпеки в безпроводових мережах підприємств»

Спеціальність: 123 «Комп’ютерна інженерія»

Освітня програма: Комп’ютерні системи захисту інформації

Керівник дипломного проекту:_____/ Лахно В.А. /
підпис ПІБ

Виконав:_____/ Івченко І.О. /
підпис ПІБ

КИЇВ-2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«ЗАТВЕРДЖУЮ»
завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
_____/ Касаткін Д.Ю., к.п.н., доцент. /
підпис ПІБ, вчене звання і ступінь
«__» _____ 2024 р.

ЗАВДАННЯ

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Івченку Івану Олеговичу
(прізвище, ім'я, по батькові)

Спеціальність (напрям підготовки): 123 «Комп'ютерна інженерія»
Освітня програма: Комп'ютерні системи захисту інформації
Тема магістерської роботи: Дослідження методів оцінювання ризиків інформаційної безпеки в безпроводових мережах підприємств
затверджена наказом ректора НУБІП України від " 1 " листопада 2021 № 1859 "С"
Термін подання завершеної роботи на кафедру _____
Вихідні дані до магістерської роботи: _____

Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області для дослідження методів оцінки ризиків
2. Огляд математичних моделей для оцінки ризиків
3. Побудова інтегрованої математичної моделі оцінки ризиків

Дата видачі завдання " 1 " вересня 2024 р.

Керівник магістерської роботи _____ / Лахно В.А. д.т.н., професор /
(підпис) (ПІБ, вчене звання і ступінь)

Завдання прийняв до виконання _____ / Івченко І.О. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка задачі магістерської роботи	01.09.2024	Виконано
2	Аналіз предметної області	10.09.2024	Виконано
3	Проєктування системи	15.09.2024	Виконано
4	Реалізація системи	05.10.2024	Виконано
5	Тестування розробленої системи	15.10.2024	Виконано
6	Оформлення пояснювальної записки	01.11.2024	Виконано
7	Оформлення графічного матеріалу	05.11.2024	Виконано

Студент _____ / Івченко І.О. /
підпис ПІБ

Керівник проєкту (роботи) _____ / Лахно В.А. /
підпис ПІБ

РЕФЕРАТ

Пояснювальна записка: 100 сторінок, 2 рисунка, 4 таблиці, 3 лістинга, 20 джерел.

РИЗИКИ, ОЦІНЮВАННЯ РИЗИКІВ, МАТЕМАТИЧНА МОДЕЛЬ, ІНФОРМАЦІЙНА БЕЗПЕКА, АЛГОРИТМИ, PYTHON

Мета – метою роботи є розробка об'єднаної математичної моделі для оцінки ризиків інформаційної безпеки в бездротових мережах підприємств, яка інтегрує сильні сторони існуючих моделей для забезпечення більш точної та гнучкої оцінки ризиків.

Об'єкт – інтерактивне хмарне середовище для програмної розробки Google Colab.

Предмет – математичні моделі оцінки ризиків інформаційної безпеки в бездротових мережах підприємств та їх інтеграція для створення об'єднаної моделі. Робота складається з трьох розділів.

У першому розділі проведений аналіз предметної області, зокрема розглянуто сучасні методи та засоби оцінювання ризиків.

У другому розділі проведено оцінку математичних моделей оцінки ризиків й принципів їх роботи, переваг та недоліків, а також ряду існуючих наукових статей, публікацій, досліджень на дану тематику.

Третій розділ є програмною реалізацією мовою Python у середовищі Google Colab інтегрованої математичної моделі, яка буде поєднувати переваги декількох існуючих.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

WIFI	–	Wireless networking technology , Бездротова мережа
LTE	–	Long-Term Evolution (network) , Тривала еволюція (мережа)
MITM	–	Man-in-the-middle attack, Атака "людина посередині"
ISO	–	International Organization for Standardization , Міжнародна організація зі стандартизації
IEC	–	International Electrotechnical Commission , Міжнародна електротехнічна комісія
NIST	–	National Institute of Standards and Technology , Національний інститут стандартів і технологій США
DSS	–	Decision support system , Система підтримки прийняття рішень
США	–	Сполучені Штати Америкids
OCTAVE	–	Operationally Critical Threat, Asset, and Vulnerability Evaluation , Оцінка оперативно критичних загроз, активів та вразливостей
RMF	–	NIST Risk Management Framework , Рамкова система управління ризиками NIST
FAIR	–	Factor Analysis of Information Risk , Факторний аналіз інформаційного ризику
ISMS	–	Information Security Management Systems , Системи управління інформаційною безпекою
SIEM	–	Security Information and Event Management , Управління інформацією та подіями безпеки
ML	–	Machine Learning , Машинне навчання
ШІ	–	Штучний Інтелект
CPD	–	Conditional Probability Distributions , Умовні ймовірнісні розподіли
DAG	–	Directed Acyclic Graph , Направлений ациклічний граф
WEP	–	Wired Equivalent Privacy , Провідний еквівалент конфіденційності
WPA	–	Wi-Fi Protected Access , Захищений доступ до Wi-Fi
МВП	–	Марковські випадкові поля

ВСТУП

Розвиток інформаційних технологій суттєво змінив підходи до організації бізнес-процесів та взаємодії між різними елементами корпоративної інфраструктури. В умовах сучасного ринку, підприємства змушені адаптуватися до нових викликів та постійно удосконалювати свої технологічні рішення. Однією з ключових інновацій, що значно полегшила і прискорила цей процес, є широке впровадження безпроводових мереж, таких як Wi-Fi, Bluetooth, LTE, 5G та інші.

Завдяки використанню безпроводових мереж підприємства можуть швидко і без значних витрат розгортати інфраструктуру для передачі даних, що забезпечує підвищену мобільність, гнучкість та масштабованість інформаційних систем. Безпроводові технології дозволяють організаціям значно спростити процеси обміну інформацією між співробітниками, забезпечити віддалений доступ до корпоративних ресурсів, а також інтегрувати різноманітні пристрої у єдину систему управління.

Незважаючи на очевидні переваги, широке застосування безпроводових мереж створює нові загрози для інформаційної безпеки підприємств. Основна відмінність безпроводових технологій від традиційних дротових рішень полягає у використанні радіочастот для передачі даних, що значно збільшує вразливість таких мереж до зовнішніх атак. Серед основних загроз варто відзначити перехоплення трафіку, несанкціонований доступ до мережі, атаки типу «людина посередині» (MITM), встановлення підроблених точок доступу, а також численні вразливості протоколів шифрування.

Забезпечення надійного рівня інформаційної безпеки в безпроводових мережах вимагає від підприємств не лише застосування технічних засобів захисту, але й побудови комплексної системи управління ризиками. Сучасна практика показує, що навіть найсучасніші засоби захисту не можуть гарантувати абсолютну безпеку, оскільки загрози постійно еволюціонують, а нові методи атак з’являються практично щодня.

Процес управління ризиками інформаційної безпеки в безпроводових мережах підприємства включає кілька етапів: ідентифікація загроз, оцінка їхнього потенційного впливу на діяльність організації, визначення ймовірності реалізації загроз, а також вибір і впровадження відповідних заходів захисту. Саме оцінка ризиків стає основою для прийняття рішень щодо впровадження тих чи інших заходів безпеки.

Важливо відзначити, що підходи до оцінки ризиків у безпроводових мережах значно відрізняються від підходів, що використовуються для дротових мереж. Це пов'язано із специфікою безпроводових технологій, зокрема їхньою схильністю до перехоплення даних та атак на фізичному рівні (наприклад, через радіочастотний інтерфейс). До того ж, мобільність користувачів безпроводових мереж ускладнює їхній захист, оскільки користувачі можуть підключатися до корпоративних систем з різних пристроїв і мереж, що підвищує ризики компрометації.

На сьогодні існує низка підходів та методик для оцінювання ризиків інформаційної безпеки в безпроводових мережах. Деякі з них базуються на стандартах та рекомендаціях міжнародних організацій, таких як ISO/IEC 27001, NIST (Національний інститут стандартів і технологій США), інші – на власних розробках компаній, які активно використовують безпроводові мережі у своїй діяльності. Однак, кожен з цих підходів має свої сильні та слабкі сторони, що впливають на точність оцінки ризиків та ефективність заходів безпеки.

Проблема оцінювання ризиків інформаційної безпеки у безпроводових мережах є надзвичайно актуальною, оскільки зростання обсягів даних, що передаються через такі мережі, та збільшення кількості пристроїв, підключених до них, значно підвищує ймовірність реалізації різноманітних загроз. Враховуючи високу вартість витоків інформації для підприємств, зокрема фінансові втрати та репутаційні ризики, розробка ефективних методів оцінювання ризиків стає критично важливою задачею.

Метою даної магістерської роботи є дослідження існуючих методів оцінювання ризиків інформаційної безпеки у безпроводових мережах підприємств.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Математичні моделі в оцінці ризиків

Інформаційна безпека є однією з ключових складових сучасного цифрового суспільства та бізнесу. З розвитком технологій зростає кількість загроз, які можуть негативно вплинути на організації та окремих користувачів. Оцінка ризиків інформаційної безпеки дозволяє ідентифікувати потенційні загрози, визначити їхню ймовірність та можливі наслідки, а також розробити ефективні стратегії захисту. У цьому процесі важливу роль відіграють математичні моделі, які забезпечують кількісний аналіз ризиків і підтримують прийняття обґрунтованих рішень.

Інформаційна безпека включає захист інформаційних систем від несанкціонованого доступу, використання, розголошення, руйнування, модифікації або порушення їхньої доступності. У сучасних умовах, коли інформація стає одним з найцінніших активів організацій, забезпечення її безпеки набуває особливої важливості. Оцінка ризиків інформаційної безпеки допомагає виявити слабкі місця систем, передбачити можливі атаки та мінімізувати їхній вплив.

Математичні моделі дозволяють систематизувати та кількісно оцінити різні аспекти ризиків інформаційної безпеки. Вони базуються на математичних методах та алгоритмах, що забезпечують об'єктивність та точність аналізу. Основні типи математичних моделей, що використовуються для оцінки ризиків, включають:

Статистичні моделі

Статистичні моделі використовують історичні дані для прогнозування майбутніх подій. В контексті інформаційної безпеки вони можуть аналізувати минулі інциденти, тенденції в кіберзлочинності, частоту та характер атак тощо. Основні методи включають регресійний аналіз, аналіз часових рядів, кластеризацію та інші статистичні техніки.

Приклад застосування: Аналіз частоти та типів кібер-атак за останні кілька років для прогнозування ймовірності подібних інцидентів у майбутньому.

Ймовірнісні моделі

Ймовірнісні моделі використовують теорію ймовірностей для оцінки невизначеності ризиків. Вони дозволяють визначити ймовірність настання певних подій та їхнього впливу на систему.

Моделі Баєса: Один із найбільш поширених підходів у ймовірнісних моделях. Вони дозволяють оновлювати ймовірності ризиків на основі нової інформації.

Марковські ланцюги: Використовуються для моделювання систем, що переходять з одного стану в інший з певними ймовірностями.

Приклад застосування: Використання моделі Баєса для оцінки ймовірності успішної фішингової атаки на організацію на основі нових даних про захист електронної пошти.

Детерміновані моделі

Детерміновані моделі використовують визначені математичні формули та алгоритми для оцінки ризиків без врахування випадкових змін. Вони базуються на заданих вхідних параметрах і забезпечують однозначний результат.

Приклад застосування: Використання моделі оцінки втрат для визначення фінансових наслідків потенційного інциденту безпеки, враховуючи витрати на відновлення, штрафи та втрату репутації.

Моделі теорії ігор

Теорія ігор застосовується для аналізу взаємодії між захисниками інформаційних систем та потенційними зловмисниками. Вона дозволяє прогнозувати поведінку атакуючих та розробляти оптимальні стратегії захисту.

Приклад застосування: Моделювання взаємодії між хакером і системою захисту для визначення оптимальних заходів безпеки, що мінімізують ймовірність успішної атаки.

Системи підтримки прийняття рішень (DSS)

Системи підтримки прийняття рішень використовують різні математичні моделі для аналізу ризиків та надання рекомендацій щодо заходів безпеки. Вони

інтегрують дані з різних джерел і забезпечують користувачам можливість аналізувати різні сценарії.

Приклад застосування: Використання DSS для вибору найбільш ефективних заходів безпеки на основі аналізу ризиків та доступних ресурсів.

1.2. Методології оцінки ризиків

Для ефективної оцінки ризиків інформаційної безпеки використовуються різні методології, які базуються на математичних моделях. Найбільш поширені з них:

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)

OCTAVE є методологією самооцінки ризиків, яка фокусується на організаційних потребах та ресурсах. Вона використовує комплексний підхід до ідентифікації та аналізу ризиків, враховуючи як технічні, так і управлінські аспекти.

Основні етапи:

- Визначення критичних активів.
- Ідентифікація загроз та вразливостей.
- Оцінка ризиків та визначення пріоритетів.
- Розробка стратегії управління ризиками.

NIST Risk Management Framework (RMF)

NIST RMF – це структурований підхід до управління ризиками, розроблений Національним інститутом стандартів і технологій США. Він включає послідовні кроки для ідентифікації, аналізу, оцінки та управління ризиками.

Основні етапи:

- Категоризація інформаційних систем.
- Вибір заходів безпеки.

- Впровадження заходів безпеки.
- Оцінка ефективності заходів.
- Моніторинг ризиків.

FAIR (Factor Analysis of Information Risk)

FAIR – це модель, яка забезпечує структурований підхід до оцінки ризиків інформаційної безпеки. Вона використовує кількісні методи для визначення ймовірності та впливу ризиків.

Основні компоненти:

- Ідентифікація факторів загрози.
- Оцінка потенційних втрат.
- Визначення ймовірності настання ризику.
- Розробка рекомендацій щодо зменшення ризиків.

1.3. Приклади застосування математичних моделей

Оцінка ризиків у банківській сфері

Банки використовують математичні моделі для оцінки ризиків, пов’язаних з фінансовими транзакціями, захистом персональних даних клієнтів та забезпеченням безпеки онлайн-банкінгу. Наприклад, модель регресії може використовуватися для прогнозування ймовірності шахрайських транзакцій на основі історичних даних.

Захист державних інформаційних систем

Державні організації використовують складні математичні моделі для захисту критичних інфраструктур, таких як енергетика, транспорт та охорона здоров’я. Моделі теорії ігор допомагають прогнозувати дії потенційних атакуючих та розробляти ефективні стратегії захисту.

Медичні інформаційні системи

У сфері охорони здоров’я захист електронних медичних записів є критично важливим. Математичні моделі допомагають оцінювати ризики, пов’язані з

доступом до конфіденційних даних, та визначати ефективні заходи безпеки для їхнього захисту.

1.4. Виклики при використанні математичних моделей

Використання математичних моделей для оцінки ризиків інформаційної безпеки приносить значні переваги, проте також супроводжується рядом викликів. Розглянемо основні з них більш детально.

Якість даних

Проблема: Якість даних є критично важливою для точності та надійності математичних моделей. Недостовірні, неповні або застарілі дані можуть призвести до неправильних висновків та неточних оцінок ризиків.

Причини:

- Неузгодженість джерел даних: Дані можуть надходити з різних джерел, які не завжди узгоджені між собою.
- Відсутність даних: Деякі важливі аспекти ризику можуть бути недостатньо задокументовані або відсутні.
- Помилки в даних: Людські помилки при введенні даних або технічні збої можуть спричинити неточності.

Рішення:

- Валідація даних: Регулярна перевірка достовірності та цілісності даних.
- Стандартизація: Використання стандартних форматів та протоколів для збору та зберігання даних.
- Автоматизація збору даних: Застосування автоматизованих систем для мінімізації людських помилок.

Складність моделей

Проблема: Деякі математичні моделі можуть бути надто складними для розуміння та впровадження, особливо для організацій, які не мають достатнього рівня математичної експертизи.

Причини:

- Високий рівень абстракції: Моделі можуть включати складні математичні концепції, які важко інтерпретувати.
- Обчислювальна складність: Вимоги до обчислювальних ресурсів можуть бути високими, що ускладнює використання моделей в реальному часі.
- Недостатня документація: Відсутність достатньої документації може ускладнити розуміння та використання моделі.

Рішення:

- Спрощення моделей: Розробка більш простих, але адекватно точних моделей.
- Навчання персоналу: Інвестування в навчання співробітників для підвищення їхніх навичок у математичному моделюванні.
- Використання інтерфейсів користувача: Розробка інтуїтивно зрозумілих інтерфейсів для роботи з моделями без необхідності глибокого розуміння їхньої внутрішньої логіки.

Динамічність загроз

Проблема: Загрози інформаційної безпеки постійно змінюються та еволюціонують, що вимагає регулярного оновлення та адаптації математичних моделей.

Причини:

- Нові види атак: Постійне виникнення нових методів атак та вразливостей.
- Зміни в технологіях: Впровадження нових технологій змінює ландшафт інформаційної безпеки.
- Регуляторні зміни: Зміни в законодавстві та стандартах можуть вимагати коригування моделей.

Рішення:

- Гнучкість моделей: Розробка адаптивних моделей, які можуть швидко інтегрувати нові дані та тенденції.
- Постійний моніторинг: Впровадження систем постійного моніторингу для виявлення змін у загрозах.
- Регулярні оновлення: Встановлення процесів для регулярного перегляду та оновлення моделей відповідно до нових даних та умов.

Взаємодія різних факторів

Проблема: Інформаційна безпека включає безліч взаємопов'язаних факторів, таких як технічні, організаційні та людські аспекти. Моделювання взаємодії цих факторів може бути складним та вимагати комплексних підходів.

Причини:

- Багатогранність ризиків: Ризики можуть мати різні джерела та впливати на різні аспекти організації.
- Складні взаємозв'язки: Взаємодія між різними факторами може бути нелінійною та непередбачуваною.
- Недостатнє розуміння: Можлива недостатня кількість досліджень, які б описували взаємозв'язки між різними факторами ризику.

Рішення:

- Системний підхід: Використання системного підходу для врахування всіх взаємопов'язаних аспектів.
- Мультимодельні системи: Інтеграція кількох моделей, кожна з яких фокусується на окремому аспекті ризику.
- Колаборативний аналіз: Співпраця між різними відділами організації для повного розуміння та врахування всіх факторів.

1.5. Переваги використання математичних моделей

Математичні моделі пропонують низку значних переваг при оцінці ризиків інформаційної безпеки, що робить їх незамінним інструментом для сучасних організацій.

Кількісна оцінка ризиків

Опис: Математичні моделі дозволяють перевести суб'єктивні оцінки ризиків у кількісні показники, що робить процес оцінки більш об'єктивним та стандартизованим.

Переваги:

- Об'єктивність: Зменшує вплив особистих упереджень на оцінку ризиків.
- Порівнянність: Забезпечує можливість порівнювати різні ризики за єдиною шкалою.
- Простота комунікації: Кількісні показники полегшують передачу інформації між різними рівнями організації.

Приклад: Використання коефіцієнтів ризику для визначення фінансових втрат від потенційних атак, що дозволяє керівництву приймати обґрунтовані рішення щодо інвестицій у заходи безпеки.

Прогнозування майбутніх загроз

Опис: Математичні моделі сприяють прогнозуванню майбутніх загроз на основі аналізу історичних даних та тенденцій.

Переваги:

- Передбачуваність: Допомагає організаціям бути проактивними у захисті від нових загроз.
- Планування: Полегшує стратегічне планування та розробку довгострокових заходів безпеки.
- Зменшення невизначеності: Знижує рівень невизначеності щодо потенційних загроз.

Приклад: Аналіз історичних даних про кібер-атаки дозволяє передбачити найбільш ймовірні напрямки майбутніх атак і відповідно коригувати заходи безпеки.

Оптимізація ресурсів

Опис: Математичні моделі допомагають визначити найбільш критичні ризики, що дозволяє ефективніше розподіляти ресурси на заходи безпеки.

Переваги:

- Ефективність: Забезпечує максимальний вплив витрат на безпеку.
- Зменшення витрат: Мінімізує витрати на непотрібні або менш ефективні заходи.
- Приоритезація: Допомагає визначити пріоритети у впровадженні заходів безпеки.

Приклад: Оптимізація бюджету на безпеку шляхом інвестування більше коштів у захист критичних систем, що мають найвищий рівень ризику.

Підтримка прийняття рішень

Опис: Математичні моделі надають об'єктивну інформацію та рекомендації, що підтримують керівників у прийнятті стратегічних рішень щодо управління ризиками.

Переваги:

- Інформованість: Забезпечує керівництво точними та своєчасними даними для прийняття рішень.
- Стратегічність: Дозволяє розробляти довгострокові стратегії безпеки на основі наукових даних.
- Зменшення ризику помилок: Мінімізує ймовірність прийняття неправильних рішень через недостатню інформацію або упередження.

Приклад: Використання математичної моделі для оцінки ефективності різних заходів безпеки та вибір найбільш ефективного варіанту для захисту інформаційних систем.

1.8. Інтеграція математичних моделей у системи управління інформаційною безпекою (ISMS)

Для ефективного використання математичних моделей необхідно інтегрувати їх у загальну систему управління інформаційною безпекою (ISMS). Це дозволяє забезпечити цілісний підхід до управління ризиками та підвищити ефективність заходів безпеки.

Вибір відповідних моделей

Опис: Вибір математичних моделей залежить від специфіки організації, її потреб та типу інформаційних активів. Важливо обрати моделі, які найбільше відповідають вимогам та особливостям конкретної організації.

Критерії вибору:

- Тип інформаційних активів: Вибір моделі залежить від того, які активи потрібно захистити (дані, системи, мережі тощо).
- Рівень ризику: Моделі повинні відповідати рівню складності та серйозності ризиків, що оцінюються.
- Ресурси організації: Врахування доступних ресурсів для впровадження та підтримки моделі.

Приклад: Для великої фінансової установи може бути обрана складна ймовірнісна модель, яка враховує різні аспекти фінансових ризиків та їх вплив на інформаційні системи.

Навчання персоналу

Опис: Для ефективного використання математичних моделей необхідно забезпечити відповідний рівень підготовки персоналу. Це включає навчання з математичного моделювання, аналізу даних та інтерпретації результатів.

Кроки досягнення:

- Проведення тренінгів: Організація регулярних тренінгів та семінарів для підвищення кваліфікації співробітників.
- Сертифікація: Заохочення персоналу до отримання сертифікатів у сфері інформаційної безпеки та математичного моделювання.
- Співпраця з експертами: Залучення зовнішніх експертів для проведення спеціалізованих навчань.

Приклад: Навчання команди безпеки основам теорії ймовірностей та статистики для кращого розуміння та застосування ймовірнісних моделей у оцінці ризиків.

Інтеграція з іншими системами

Опис: Математичні моделі повинні бути інтегровані з іншими системами управління інформаційною безпекою, такими як системи моніторингу, виявлення загроз та реагування на інциденти.

Переваги інтеграції:

- Єдність даних: Забезпечує обмін даними між різними системами для більш точного аналізу.
- Автоматизація процесів: Дозволяє автоматизувати деякі аспекти оцінки ризиків та реагування на загрози.
- Покращена ефективність: Підвищує загальну ефективність системи управління безпекою.

Приклад: Інтеграція математичної моделі з системою SIEM (Security Information and Event Management) для автоматичного оновлення оцінок ризиків на основі реального часу даних про безпеку.

Постійний моніторинг та оновлення

Опис: Оцінка ризиків є динамічним процесом, що вимагає постійного моніторингу та оновлення моделей на основі нових даних та змін у середовищі загроз.

Кроки досягнення:

- Регулярний перегляд моделей: Встановлення графіку для регулярного перегляду та оновлення моделей.
- Автоматизовані системи моніторингу: Використання систем, які автоматично збирають та аналізують нові дані.
- Аналіз змін: Виявлення змін у зовнішньому та внутрішньому середовищі, що можуть вплинути на оцінку ризиків.

Приклад: Щоквартальне оновлення математичної моделі на основі нових даних про кібер-атаки та зміни у технологічному середовищі організації.

1.9. Майбутні тенденції у використанні математичних моделей

Сфера інформаційної безпеки швидко розвивається, і математичні моделі також зазнають значних змін та покращень. Розглянемо основні майбутні тенденції.

Використання штучного інтелекту та машинного навчання

Опис: Штучний інтелект (ШІ) та машинне навчання (ML) відкривають нові можливості для оцінки ризиків інформаційної безпеки. Вони дозволяють автоматизувати аналіз великих обсягів даних, виявляти складні патерни та прогнозувати загрози з високою точністю.

Переваги:

- Автоматизація: Зменшує потребу в ручному аналізі даних, що скорочує час та знижує витрати.
- Покращена точність: ML алгоритми можуть виявляти аномалії та патерни, які важко виявити традиційними методами.
- Адаптивність: Моделі можуть навчатися на нових даних та вдосконалювати свої прогнози з часом.

Приклад: Використання нейронних мереж для аналізу трафіку мережі та виявлення підозрілих активностей, що можуть свідчити про кібер-атаки.

Інтеграція з великими даними (Big Data)

Опис: Зростання обсягів даних вимагає використання математичних моделей, здатних ефективно аналізувати великі набори даних. Інтеграція з Big Data дозволяє отримувати більш детальну та точну оцінку ризиків.

Переваги:

- Масштабованість: Моделі можуть обробляти великі обсяги даних без значного зниження продуктивності.
- Глибший аналіз: Дозволяє аналізувати дані з різних джерел для більш повного розуміння ризиків.
- Швидкість: Швидкий аналіз великих обсягів даних дозволяє оперативно реагувати на загрози.

Приклад: Аналіз великих обсягів логів серверів та мережевих пристроїв для виявлення тенденцій та аномалій, які можуть вказувати на потенційні загрози.

Розвиток квантових обчислень

Опис: Квантові обчислення можуть значно підвищити швидкість та ефективність математичних моделей, дозволяючи вирішувати складніші задачі та обробляти більші обсяги даних.

Переваги:

- Підвищена обчислювальна потужність: Можливість вирішення задач, що є недосяжними для класичних комп'ютерів.
- Оптимізація алгоритмів: Квантові алгоритми можуть оптимізувати процеси оцінки ризиків та аналізу даних.
- Покращена безпека: Квантові методи можуть забезпечити нові рівні захисту інформації.

Приклад: Використання квантових алгоритмів для оптимізації розподілу ресурсів у системах інформаційної безпеки, забезпечуючи максимальну ефективність при мінімальних витратах.

Адаптивні та самонавчальні моделі

Опис: Розробка адаптивних та самонавчальних моделей дозволяє їм автоматично оновлюватися та вдосконалюватися на основі нових даних, забезпечуючи більш точну та актуальну оцінку ризиків.

Переваги:

- Самооптимізація: Моделі можуть самостійно вдосконалювати свої алгоритми на основі нових даних.
- Гнучкість: Дозволяє швидко адаптуватися до змін у середовищі загроз.
- Підвищена точність: Постійне навчання на нових даних покращує точність прогнозів.

Приклад: Використання самонавчальної моделі, яка аналізує нові типи атак та автоматично адаптує свої алгоритми для ефективного виявлення цих атак у майбутньому.

2 ПРАКТИЧНА ЧАСТИНА

2.1 Баєсівська модель оцінки ризиків

В цій частині диплому ми здійснимо огляд математичних моделей оцінювання ризиків, а також здійснимо огляд їх переваг та недоліків.

Огляд математичних моделей оцінювання ризиків є важливою складовою дослідження, оскільки він дозволяє ознайомитися з існуючими підходами, оцінити їх сильні та слабкі сторони, а також обрати найбільш відповідний метод для задач в області інформаційної безпеки. Для безпроводових мереж підприємств, де існують специфічні загрози, такі як перехоплення даних та несанкціонований доступ, ефективне оцінювання ризиків є необхідним для захисту інформації та забезпечення стабільності роботи мережі.

В нашій роботі ми розглянемо наступні математичні моделі:

- Баєсівська модель
- Марковські мережі

Байєсівська мережа (Bayesian Network) — це графічна модель, яка використовується для представлення та аналізу ймовірнісних залежностей між різними змінними. Вона складається з двох основних компонентів: структури (спрямованого ациклічного графа) та умовних ймовірнісних розподілів (Conditional Probability Distributions, CPD). Ця модель дозволяє ефективно обробляти та інтерпретувати складні взаємозв'язки між різними факторами ризику, що є критично важливим для оцінки інформаційної безпеки безпроводових мереж підприємств.

Основні компоненти Байєсівської мережі:

- Структура мережі (Directed Acyclic Graph, DAG)

Структура Байєсівської мережі представляється у вигляді спрямованого ациклічного графа (DAG), де:

Вузли (Nodes): Кожен вузол у графі представляє певну змінну або фактор ризику. У контексті інформаційної безпеки безпроводових мереж це можуть бути такі змінні, як рівень шифрування, сила паролів, наявність політик безпеки тощо.

Дуги (Edges): Спрямовані дуги між вузлами відображають умовні залежності між змінними. Наприклад, дуга від вузла "Рівень шифрування" до вузла "Ймовірність перехоплення даних" вказує на те, що ймовірність перехоплення залежить від рівня шифрування.

Ациклічність: В графі відсутні цикли, що забезпечує коректність ймовірнісних обчислень та дозволяє уникнути нескінченних зворотних зв'язків між змінними.

- Умовні ймовірнісні розподіли (Conditional Probability Distributions, CPD)

Кожен вузол мережі має пов'язаний з ним умовний ймовірнісний розподіл (CPD), який визначає ймовірність кожного можливого значення змінної з урахуванням значень її батьківських вузлів. CPD є ключовим елементом, що дозволяє мережі моделювати складні взаємозв'язки та забезпечує основу для ймовірнісних висновків.

Приклад умовного розподілу ймовірностей:

Таблиця 1 Ймовірність перехоплення даних залежно від рівня шифрування:

Рівень шифрування	**Ймовірність перехоплення**
WEP	0.9
WPA	0.6
WPA2	0.3
WPA3	0.1

Особливості CPD:

- Випадки без батьківських вузлів: Якщо вузол не має батьків, CPD представляє безпосередні ймовірності для кожного можливого значення змінної.
- Випадки з батьківськими вузлами: Якщо вузол має батьківські вузли, CPD визначає ймовірність кожного значення змінної для кожної можливої комбінації значень батьківських вузлів.

Математичне визначення CPD:

Для будь-якого вузла X з батьками Pa(X), умовний розподіл ймовірностей можна записати як:

$$P(X|Pa(X)) = \prod_{i=1}^{n} P(X|Pa_i(X))$$

де $Pa_i(X)$— окрема комбінація значень батьківських вузлів.

Математичні основи Байєсівської мережі

Байєсівські мережі базуються на теоремі Баєса, яка дозволяє оновлювати ймовірнісні оцінки змінних при отриманні нової інформації. Основна формула теореми Баєса виглядає наступним чином:

$$P(A|B)=\frac{p(B|A)*P(A)}{P(B)}$$

де:

P(A|B)— умовна ймовірність події A при умові B.

P(B|A)— умовна ймовірність події B при умові A.

P(A) та P(B)— ймовірності подій A та B відповідно.

Ця формула дозволяє моделювати процес оновлення ймовірностей при отриманні нової інформації, що є основним принципом роботи Байєсівських мереж.

Ця формула дозволяє моделювати процес оновлення ймовірностей при отриманні нової інформації, що є основним принципом роботи Байєсівських мереж.

В графічному вигляді, ми можемо показати цю мережу як:

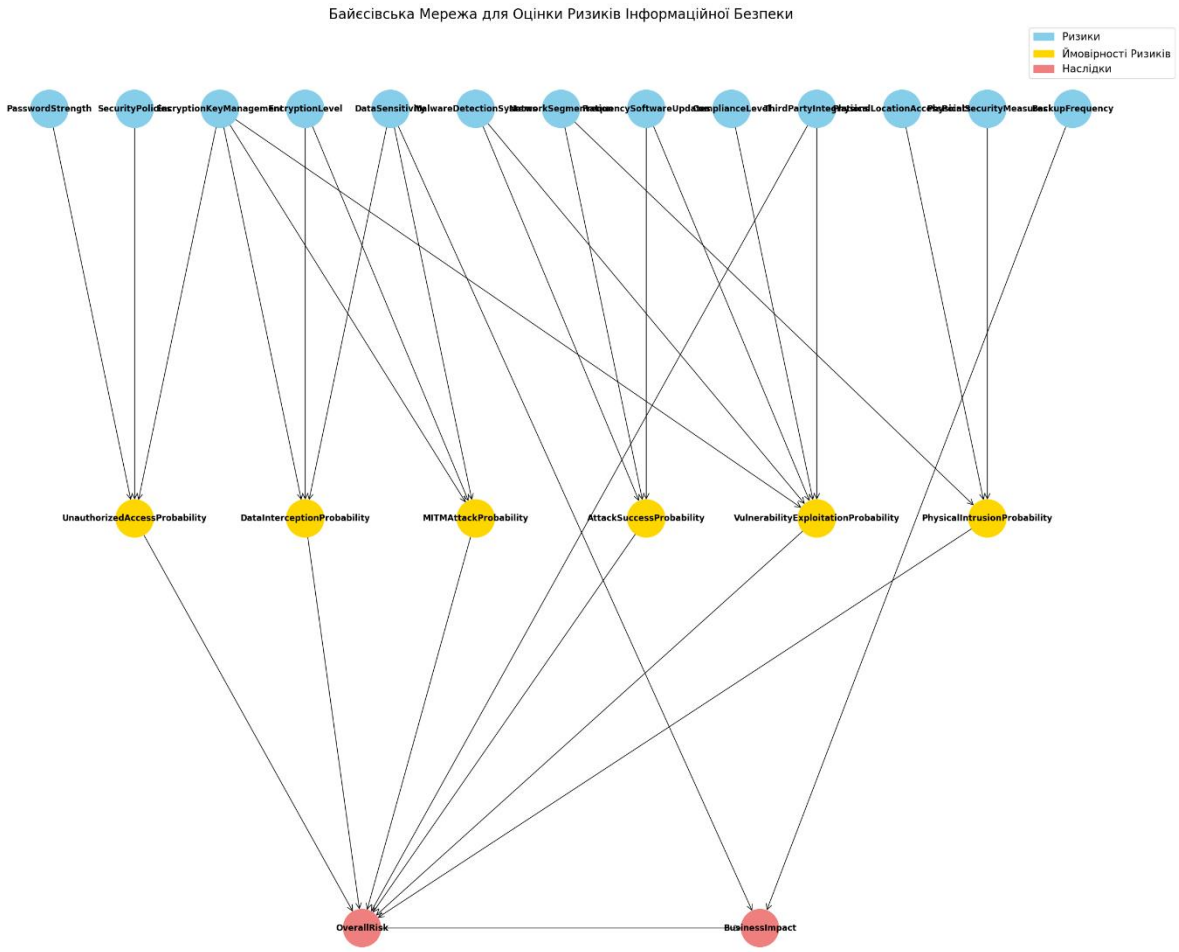

Рисунок 2.1 Приклад графічного відображення Баєсівської мережі

Переваги баєсівських мереж:

1. **Врахування невизначеності**: Баєсівські мережі дозволяють обробляти невизначеність і ймовірнісні зв’язки між подіями. Це дуже важливо в ситуаціях, де інформація неповна або є ймовірнісні залежності між різними факторами ризику. Вони дозволяють моделювати ситуації, де дані можуть бути неповними або наближеними.

2. **Аналіз залежностей між змінними**: Баєсівські мережі добре підходять для вивчення залежностей між змінними. Завдяки побудові графів, можна легко побачити, які змінні впливають одна на одну, і як зміна однієї змінної впливатиме на інші. Це дозволяє створити глибше розуміння природи ризиків та способів їх взаємодії.

3. **Можливість оновлення інформації**: У баєсівських мережах нові дані можуть бути легко додані для оновлення ймовірностей подій. Це дозволяє

адаптувати модель до нової інформації, яка надходить з часом, і забезпечує її актуальність у процесі аналізу ризиків.

4. **Підтримка прийняття рішень**: Оскільки баєсівські мережі допомагають обчислювати ймовірність настання певної події або комбінації подій, вони можуть слугувати основою для прийняття рішень. Вони забезпечують кількісну оцінку ризиків, що може допомогти виявити найбільш критичні фактори для управління ризиками.

5. **Гнучкість та адаптивність**: Ці мережі можуть застосовуватися до широкого кола завдань і налаштовуватися під конкретні потреби. Їх можна використовувати для моделювання як великих, так і малих систем з різними взаємозалежностями, що робить їх досить універсальним інструментом.

Недоліки баєсівських мереж

1. **Складність у побудові**: Створення баєсівської мережі вимагає значних аналітичних зусиль, особливо для великої кількості змінних. Щоб визначити структуру мережі і залежності між змінними, необхідно мати значну кількість даних і розуміння природи кожної змінної.

2. **Потреба у великому обсязі вхідних даних**: Для того щоб побудувати точну модель, необхідно мати достатній обсяг вхідних даних, які б відображали ймовірності та залежності між змінними. Якщо даних мало або вони неякісні, результати моделювання можуть бути неточними.

3. **Обмеженість у часі**: Баєсівські мережі можуть бути обчислювально складними, особливо якщо йдеться про велику кількість змінних і залежностей між ними. Це може уповільнювати процес оцінки ризиків у режимі реального часу, особливо якщо необхідне постійне оновлення ймовірностей.

4. **Складність інтерпретації великих моделей**: Якщо мережа має багато змінних і зв’язків, інтерпретація результатів може бути складною. Велика кількість взаємодій між факторами може зробити модель заплутаною, і потрібно ретельно аналізувати, щоб не пропустити критичні аспекти оцінки ризиків.

5. **Відсутність універсальних стандартів побудови**: У той час як баєсівські мережі є потужним інструментом, їх налаштування часто залежить від конкретної проблемної області. Це означає, що існує багато варіантів побудови і

підходів, що може створювати труднощі для стандартизації процесу моделювання.

Отже, баєсівські мережі є ефективними для моделювання систем з багатьма взаємопов’язаними факторами і невизначеностями, але вимагають значних ресурсів для створення і підтримки. У випадках, коли дані є обмеженими або модель має бути швидко адаптована, цей підхід може виявитися складним.

Розглянемо також інші моделі.

2.2 Марковські мережі

Марковські випадкові поля (МВП), або Марковські мережі, є ненаправленими графічними моделями, що використовуються для моделювання залежностей між змінними у системах, які можуть зазнавати випадкових змін. МВП широко застосовуються в задачах оцінки ризиків, обробки зображень, обчислювальної біології та кібербезпеки. Їхня структура дозволяє представляти ймовірнісні зв’язки між змінними, які не обов'язково є причинно-наслідковими, а демонструють певні зв’язки або взаємодії між факторами.

Марковські мережі відрізняються від байєсівських мереж тим, що вони використовують ненаправлені зв’язки між змінними, а не спрямовані, що робить їх гнучкими в застосуванні до систем, де відносини між змінними важливі для аналізу, але не мають суворої спрямованості. Це особливо важливо в кібербезпеці, де різні аспекти безпеки можуть бути пов'язані один з одним без явного причинного зв’язку.

Основи теорії Марковських мереж

Марковська мережа може бути представлена графом G=(V,E), де:

- V — множина вузлів, які представляють випадкові змінні,
- E — множина ненаправлених ребер, що вказують на залежність між змінними.

Кожна змінна Xi∈V має множину станів, і залежність між змінними представлена факторною функцією ϕ, яка призначає вагу кожному можливому набору значень для групи змінних, що взаємодіють у межах локальної частини графу.

Марковська властивість

Марковські мережі ґрунтуються на марковській властивості, яка стверджує, що кожна змінна Xi є умовно незалежною від усіх інших змінних, окрім безпосередніх сусідів. Це означає, що для кожної змінної в мережі ймовірність її стану залежить лише від станів змінних, з якими вона має спільне ребро.

Формально це можна записати як:

$$P(X_i \mid X_{V\setminus\{i\}}) = P(X_i \mid X_{neigh(Xi)})$$

де neigh(Xi)— сусіди вузла Xi.

Спільний розподіл ймовірностей

У марковських мережах спільний розподіл ймовірностей усіх змінних може бути представлений через факторні функції:

$$P(X) = \frac{1}{Z} \prod_{C \in C} \varphi_C(X_C)$$

де:

- X — сукупність всіх змінних у моделі,
- C — множина клік (повних підграфів) графа G,
- φ_C (X_C)— факторна функція для кожної кліки C,
- Z — нормалізаційна константа, яка обчислюється як сума всіх можливих значень факторів для всіх змінних, тобто:

$$Z = \sum_X \prod_{C \in C} \varphi_C\,(X_C)$$

Нормалізаційна константа Z гарантує, що всі ймовірності у системі сумуються до одиниці.

Кроки побудови моделі

1. **Визначення вузлів (змінних)**: Для кожного аспекту безпеки (наприклад, частота оновлень, політики безпеки, рівень шифрування) створюється вузол зі списком можливих станів (наприклад, "низький", "середній", "високий").

2. **Встановлення залежностей (ребер)**: Між змінними додаються ненаправлені зв’язки, які вказують на залежності між факторами. Наприклад, політика безпеки та частота оновлень можуть впливати на ймовірність успіху атаки.

3. **Створення факторних функцій**: Для кожної групи пов’язаних змінних створюється факторна функція ϕ\phiϕ, яка визначає ваги для різних комбінацій станів. Це дозволяє моделі відображати взаємний вплив змінних на ймовірність ризикових подій.

4. **Інференція (висновки)**: Використання алгоритмів, таких як Variable Elimination, дозволяє обчислювати умовні ймовірності для окремих змінних з урахуванням поточного сценарію доказів.

Інференція та обчислення ймовірностей

У марковській мережі обчислення ймовірностей різних станів змінних здійснюється за допомогою інференції. Для цього застосовуються алгоритми, такі як усунення змінних (Variable Elimination) та розповсюдження переконань (Belief Propagation), які дозволяють обчислити ймовірність конкретного стану для змінних на основі встановлених доказів.

Цей процес дозволяє обчислити ймовірність успіху атаки за різних налаштувань безпеки.

Марковські мережі є потужним інструментом для моделювання залежностей між факторами ризику в кібербезпеці, дозволяючи враховувати вплив змінних одна на одну та розраховувати ймовірності виникнення ризикових подій. Завдяки їхній здатності моделювати складні взаємозв’язки, вони забезпечують більш точну оцінку ризиків, що дозволяє ухвалювати обґрунтовані рішення щодо посилення безпеки і мінімізації потенційних збитків.

Переваги Марковських мереж (Марковських випадкових полів) у контексті оцінки ризиків

1. **Динамічне моделювання**: Марковські мережі добре підходять для систем, де можливі часті та динамічні зміни станів. Вони дозволяють не лише оцінювати поточний стан системи, але й прогнозувати розвиток подій, що є важливим у кібербезпеці для виявлення потенційних загроз.

2. **Врахування складних взаємозалежностей**: Завдяки своїй структурі, марковські мережі можуть враховувати взаємозалежності між численними факторами ризику, не вимагаючи при цьому жорсткої спрямованості залежностей (як у байєсівських мережах). Це робить їх корисними для моделювання складних систем без обов’язкової причинно-наслідкової залежності.

3. **Гнучкість у побудові моделей**: Можливість встановлення ненаправлених зв’язків дає змогу створювати гнучкі моделі, які можна адаптувати до різних сценаріїв і систем. Це дозволяє легко змінювати структуру моделі та додавати нові змінні за потреби.

4. **Інструмент для оцінки ризику в умовах невизначеності**: Марковські мережі можуть ефективно моделювати ймовірнісні залежності навіть у тих випадках, коли не всі змінні мають точні значення. Це робить їх придатними для оцінки ризиків, де є елементи невизначеності або нечіткі значення.

5. **Прогнозування та обґрунтування прийняття рішень**: Завдяки можливості оцінки ймовірностей переходів між станами, марковські мережі дають змогу прогнозувати розвиток ризиків і ухвалювати обґрунтовані рішення щодо впровадження заходів безпеки.

Недоліки Марковських мереж

1. **Висока вимога до обсягу даних**: Для побудови точних моделей марковської мережі потрібен великий обсяг даних, щоб правильно оцінити ймовірності переходів між станами. Недостатній обсяг даних може призвести до неточних результатів, що знижує ефективність моделі.

2. **Складність моделювання великих систем**: При моделюванні великих систем із багатьма змінними та зв’язками, марковські мережі стають обчислювально складними. Збільшення кількості змінних призводить до експоненціального зростання обчислювальної складності, що може ускладнити роботу з моделлю.

3. **Відсутність "пам'яті" про попередні стани**: Марковська мережа базується на марковській властивості, яка передбачає, що кожен новий стан залежить тільки від поточного, а не від усіх попередніх станів. Це обмежує

можливість моделювання систем, де важливий весь ланцюг переходів або накопичення ризиків із часом.

4. **Інтерпретаційна складність великих моделей**: У складних мережах з великою кількістю змінних важко інтерпретувати результати інференції, що може ускладнити прийняття рішень. Розуміння всіх залежностей та їхнього впливу може вимагати додаткового часу та аналітичних ресурсів.

5. **Необхідність точного налаштування факторів та потенціалів**: Щоб модель була ефективною, необхідно ретельно налаштувати фактори ймовірностей для кожної змінної. Неправильне налаштування може призвести до неправильних прогнозів ризиків, а в багатьох випадках таке налаштування потребує експертного аналізу та додаткових даних.

2.3 Нечіткі мережі (fuzzy-networks)

Нечіткі мережі (або нечіткі мережеві моделі) є потужним інструментом для моделювання та обробки систем, що працюють в умовах невизначеності. Основою цього підходу є нечітка логіка, яка дозволяє працювати з поняттями, що не мають чітко визначених значень, але можуть бути описані лінгвістичними змінними, наприклад, такими як "низький ризик", "середній ризик" або "високий ризик". Нечіткі мережі використовуються у випадках, коли традиційні ймовірнісні підходи не можуть дати точного результату, оскільки вони залежать від імовірностей, які можуть бути недоступними або ненадійними.

Нечітка логіка дозволяє ефективно моделювати ситуації, де значення змінних мають розмиті межі, а нечіткі мережі надають можливість побудови моделей, де змінні взаємодіють між собою через нечіткі правила, а не строго детерміновані залежності.

Основи теорії нечітких мереж

Нечіткі мережі використовують поняття нечітких множин для представлення значень змінних, які можуть бути лише приблизно визначеними.

Кожна нечітка змінна характеризується функцією належності μ(x)\mu(x)μ(x), яка відображає ступінь належності значення xxx до певної нечіткої множини. Ступінь належності завжди знаходиться в інтервалі [0,1][0, 1][0,1], де:

- μ(x)=1 означає, що значення повністю належить нечіткій множині,
- μ(x)=0 означає, що значення не належить множині,
- 0<μ(x)<1 вказує на часткове належання.

Формальна структура нечітких мереж

Нечітка мережа може бути представлена як сукупність нечітких правил і зв'язків між змінними. Кожне правило містить передумову (якщо-умова) та висновок (тоді-умова), що моделює залежність між змінними у нечіткій формі.

Наприклад, нечітке правило може виглядати так:

- **Якщо** частота оновлень є "низькою" **і** рівень шифрування є "середнім", **тоді** ймовірність успіху атаки є "високою".

Нечіткі правила та висновки

Нечіткі мережі зазвичай базуються на системі нечітких правил, які дозволяють моделювати залежності між змінними. Нечітке правило має вигляд:

Якщо (умова 1) I (умова 2) I ... I (умова n), тоді (результат)

де кожна умова описується нечіткою множиною, а результат — ступенем належності до певного рівня ризику або іншого показника.

Основні кроки моделювання нечітких мереж

1. **Визначення лінгвістичних змінних**: Спершу визначаються змінні, що описують систему (наприклад, частота оновлень, рівень шифрування, політика безпеки) та їхні нечіткі значення (наприклад, "низький", "середній", "високий").

2. **Побудова функцій належності**: Кожному нечіткому значенню змінної призначається функція належності. Наприклад, функції належності можуть мати трикутну форму або бути сигмоїдальними.

3. **Формулювання нечітких правил**: Створюється набір правил, які описують залежності між змінними, наприклад:

   - **Якщо** політика безпеки є "середньою" **і** рівень шифрування "низький", **тоді** ризик несанкціонованого доступу "високий".

4. **Агрегація правил**: Виконується нечітке об'єднання результатів усіх правил для отримання узагальненого висновку. Для цього використовуються операції типу мінімуму, максимуму або середнього значення для кожної нечіткої змінної.

5. **Дефазифікація**: На останньому етапі результат нечіткої моделі переводиться у чітке значення, що відображає рівень ризику. Для цього можуть використовуватися різні методи, як-от метод центру ваги (Center of Gravity, COG).

Формули для нечітких мереж

1. **Функція належності**: Визначає ступінь належності змінної до нечіткої множини. Наприклад, для трикутної функції належності:

$$\mu(x)=\begin{cases} 0, if\ x\ \leq a\ or\ x\ \geq c \\ \frac{x-a}{b-a}\ , if\ a\ \leq x\ \leq b \\ \frac{c-x}{c-b}\ , if\ b\ \leq x\ \leq c \end{cases}$$

де a, b, c — параметри, що визначають форму функції належності.

2. **Агрегація правил**: У нечітких мережах агрегація результатів виконується шляхом застосування операції максимуму або середнього значення. Наприклад, для двох умов:

$$\mu_{результат} = \max (\mu_{умова1} , \mu_{умова2})$$

де $\mu_{результат}$ — значення ступеня належності висновку.

3. **Дефазифікація**: Для отримання чіткого результату використовується метод центру ваги:

$$x_{дефаз} = \frac{\sum_i x_{i*}\, \mu(x_i)}{\sum_i \mu(x_i)}$$

де x_i — дискретне значення вихідної змінної, а $\mu(x_i)$ — відповідна функція належності.

Приклад нечіткої мережі для оцінки ризику в кібербезпеці

У контексті кібербезпеки нечітка мережа може містити такі змінні, як:

- **Частота оновлення програмного забезпечення** (наприклад, "низька", "середня", "висока"),
- **Політика безпеки** (наприклад, "обмежена", "часткова", "повна"),
- **Рівень шифрування** (наприклад, "WPA", "WPA2", "WPA3").

На основі цих змінних формулюються нечіткі правила, які визначають ймовірність виникнення загрози. Наприклад:

- Якщо частота оновлень є "низькою" та рівень шифрування є "WPA", тоді ризик перехоплення даних є "високим".

Процедура розрахунку в нечіткій мережі

1. **Введення даних**: Задаються значення для кожної змінної. Наприклад, частота оновлень встановлена на 20% від максимальної, а політика безпеки на 70%.
2. **Обчислення функцій належності**: За допомогою відповідних функцій належності визначаються ступені належності введених значень до кожної нечіткої множини, наприклад, ступінь належності частоти оновлень до "низької", "середньої" або "високої" категорій.
3. **Застосування нечітких правил**: Виходячи з функцій належності, кожне правило оцінюється для отримання ступеня належності результату. Наприклад, ризик перехоплення даних може виявитися частково "високим" та частково "середнім".
4. **Агрегація та дефазифікація**: Результати всіх правил комбінуються для отримання загальної оцінки ризику, а дефазифікація дозволяє отримати точне числове значення, що відображає ймовірність ризику.

Переваги нечіткої логіки для оцінки ризиків

1. **Адаптивність до невизначеності**: Нечітка логіка ефективно працює в умовах невизначеності, дозволяючи моделювати ситуації, де дані є приблизними або нечіткими. Це важливо для оцінки ризиків, коли немає доступу до точних числових значень або ймовірнісних оцінок.
2. **Лінгвістичні змінні**: Нечітка логіка дозволяє використовувати лінгвістичні оцінки, такі як "низький", "середній" або "високий", які легше зрозуміти та інтерпретувати, особливо для фахівців, які не є експертами в математичному моделюванні.
3. **Гнучкість та простота налаштування**: Нечітка логіка легко налаштовується під конкретні задачі. Нечіткі правила можна формулювати

на основі експертного досвіду, що дозволяє адаптувати систему під специфічні умови або потреби.

4. **Моделювання складних взаємозв’язків**: Завдяки використанню нечітких правил, система може моделювати складні взаємозв'язки між змінними. Наприклад, для оцінки ризику можна врахувати одночасний вплив декількох факторів (частота оновлень, політика безпеки, рівень шифрування).
5. **Застосовність у реальному часі**: Нечіткі системи можуть швидко обчислювати результати і тому підходять для оцінки ризиків у режимі реального часу. Це особливо корисно для систем безпеки, де швидке реагування на потенційні загрози має вирішальне значення.

Недоліки нечіткої логіки для оцінки ризиків

1. **Суб’єктивність у формулюванні правил**: Оскільки нечіткі правила зазвичай створюються на основі експертних знань, модель може бути упередженою або занадто суб’єктивною. Це може призвести до недооцінки або переоцінки ризиків, залежно від експертного досвіду та суджень.
2. **Відсутність об’єктивної статистичної основи**: Нечітка логіка не враховує об’єктивні ймовірнісні дані, що може бути недоліком, якщо точні дані про ризики доступні. Це обмежує застосування нечіткої логіки у випадках, де важливі точні статистичні оцінки.
3. **Складність налаштування функцій належності**: Для кожної змінної необхідно визначити функції належності, які можуть бути складними для налаштування та можуть потребувати досвіду і знань специфіки проблемної області. Неправильна конфігурація функцій належності може спотворити результати.
4. **Інтерпретаційні труднощі з великою кількістю змінних**: Якщо модель включає велику кількість факторів, кількість нечітких правил значно зростає, що ускладнює її інтерпретацію та управління. Це може обмежувати застосовність нечіткої логіки в дуже складних системах.
5. **Складність дефазифікації**: Після отримання нечіткого результату необхідно виконати дефазифікацію (перетворення нечіткого значення у

чітке), яка іноді може бути важкою або може втратити частину інформації, особливо якщо результат є дуже нечітким.

2.4 Дерева рішень

Дерева рішень (Decision Trees) — це метод ієрархічного прийняття рішень, який використовується для оцінки ризиків та вибору оптимальних дій в умовах невизначеності. Це графічна модель, що представляє послідовність рішень і наслідків, побудована у вигляді дерева, де кожна гілка відповідає певній умові, а кожен листок — результату або рівню ризику. Дерева рішень допомагають візуалізувати процес оцінки ризиків, а також дають можливість побудувати структурований та інтуїтивно зрозумілий шлях до кінцевого результату.

Основні компоненти дерева рішень

1. **Кореневий вузол (Root Node)**: Початковий вузол дерева, що представляє основне питання або початковий критерій, на основі якого відбувається подальше розгалуження.
2. **Вузли умов (Internal Nodes)**: Вузли, що представляють умови або фактори ризику. Кожен вузол перевіряє конкретну змінну та спрямовує подальший шлях на основі того, чи виконується умова.
3. **Гілки (Branches)**: Гілки, що відходять від вузлів умов і представляють варіанти рішень або наслідки залежно від того, чи виконуються умови.
4. **Листки (Leaf Nodes)**: Кінцеві вузли дерева, які представляють можливі результати або рівні ризику (наприклад, "високий ризик", "низький ризик" або конкретне рішення).
5. **Класифікація або регресія**: Залежно від задачі, дерево може давати класифікацію (високий або низький ризик) або значення ризику (наприклад, ймовірність атаки).

Побудова дерева рішень для оцінки ризиків

Побудова дерева рішень включає вибір критеріїв, які найбільше впливають на ризик, і поступове розділення даних на підгрупи, де кожен вузол намагається розмежувати сценарії за ризиком.

1. **Вибір змінних (факторів ризику)**: Перший крок у побудові дерева — визначення змінних, що мають найбільший вплив на ризик. Наприклад, для бездротових мереж такими змінними можуть бути частота оновлень, тип шифрування, політика доступу, наявність систем виявлення вторгнень.
2. **Вибір роздільних умов**: Для кожної змінної визначаються умови, які дозволяють розділити дані. Наприклад, умовою може бути "Частота оновлень: низька або висока". Кожна умова розділяє вузол на дві або більше гілок.
3. **Розрахунок інформативності (Gain або Impurity)**: Використовуються показники, такі як ентропія або індекс Джині, щоб визначити, наскільки ефективно кожен вузол розділяє дані. Цей розрахунок допомагає обрати оптимальні умови для кожного розділення, які найбільш знижують невизначеність або ризик.
4. **Побудова гілок і вузлів**: Для кожного вузла перевіряються умови, і для кожного варіанту створюється новий вузол. Так формується ієрархічна структура, що представляє процес прийняття рішень.
5. **Кінцеві результати (листки)**: На завершення побудови дерева, кожен листок відображає підсумковий ризик або рішення для заданого сценарію. Це дозволяє чітко визначити кінцевий рівень ризику залежно від обраного шляху.

Переваги дерев рішень для оцінки ризиків

1. **Інтуїтивна візуалізація**: Дерева рішень забезпечують чітку візуалізацію процесу прийняття рішень. Кожен рівень дерева показує, як певні фактори впливають на кінцевий ризик, що робить процес оцінки зрозумілим для фахівців з кібербезпеки та керівництва.
2. **Простота використання**: Дерева рішень легко будувати та інтерпретувати. Вони не потребують складних обчислень і підходять для задач, де необхідно швидко оцінити ризик на основі декількох критеріїв.

3. **Визначення ключових факторів**: Дерева рішень дозволяють легко визначити, які змінні або умови мають найбільший вплив на рівень ризику. Це допомагає зосередитися на найважливіших аспектах безпеки.
4. **Адаптивність**: Дерева рішень можуть легко оновлюватися, додаючи нові вузли або змінюючи умови в існуючих вузлах.

Недоліки дерев рішень

1. **Схильність до переобучення (overfitting)**: Дерева рішень можуть бути занадто деталізованими для невеликих обсягів даних, що призводить до зниження точності на нових даних. Це особливо критично для систем кібербезпеки, де можуть з'являтися нові загрози, не враховані в навчальних даних.
2. **Обмежена здатність моделювати складні взаємозв'язки**: У випадку багатофакторних ризиків, де між змінними є складні взаємозалежності, дерева рішень можуть бути недостатньо точними або інформативними. Наприклад, важко моделювати нелінійні взаємозв'язки між параметрами безпеки.
3. **Чутливість до змін у даних**: Зміни в даних або змінні з великою кількістю значень можуть значно вплинути на структуру дерева, змінюючи умови розгалуження. Це може призвести до необхідності частої перебудови дерева.
4. **Необхідність оптимізації**: Величезні дерева рішень можуть бути важкими для інтерпретації та управління. У таких випадках потрібно застосовувати методи оптимізації (наприклад, обрізання дерева), щоб зменшити його розмір і підвищити узагальнювальну здатність.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Вступ до практичної частини

В цій частині , проаналізувавши моделі , ми створимо інтеграційну модель, яка буде більш ефективно оцінювати ризики. Для 2 моделей які ми будемо інтегрувати я обрав – Баєсівські мережі та Марковські мережі. Обидва підходи мають свої сильні сторони та обмеження. Баєсовські мережі добре підходять для моделювання явних причинно-наслідкових зв'язків, але можуть бути менш ефективними у випадках, коли залежності між змінними мають складну структуру або є симетричними. Марковські випадкові поля, зі свого боку, ефективно моделюють локальні залежності, але не забезпечують явного представлення причинності. Інтеграція цих двох підходів дозволяє об'єднати їхні переваги, створюючи модель, яка здатна як моделювати складні причинно-наслідкові зв'язки, так і враховувати локальні взаємодії між змінними.

3.2 Визначення тестового сценарію

Для тестування наших мереж ми створимо тестовий сценарій, для цього ми визначимо змінні, на базі яких ми будемо проводити аналіз а також їх можливі стани. В таблиці 1 наведено інформацію про наші змінні.

Таблиця 2 Змінні для тестового сценарію

Змінні для нашої моделі	Вплив
Рівень шифрування (Encryption Level)	Ймовірність перехоплення зменшується зі збільшенням рівня шифрування.
Сила паролів (Password Strength)	Впливає на ймовірність злому паролю.
Наявність політик безпеки (Security Policies)	Визначає рівень організаційної захищеності.
Фізичне розташування точок доступу (Physical Location of Access Points)	Захищене розташування зменшує ризик фізичних атак.
Частота оновлень програмного забезпечення (Frequency of Software Updates)	Часті оновлення покращують захист від нових загроз.
Ймовірність атак типу "людина посередині" (MITM Attack Probability)	Вища ймовірність атаки збільшує ризик перехоплення та модифікації даних.

Рівень відповідності нормативним вимогам (Compliance Level)	Високий рівень відповідності знижує ймовірність інцидентів безпеки завдяки дотриманню стандартів та процедур.
Чутливість даних (Data Sensitivity)	Вища чутливість даних вимагає більш строгих заходів безпеки, знижуючи ймовірність витоку інформації.
Сегментація мережі (Network Segmentation)	Повна сегментація мережі обмежує поширення атак, знижуючи загальний ризик.
Системи виявлення шкідливого програмного забезпечення (Malware Detection Systems)	Високий рівень виявлення знижує ймовірність успішних атак шкідливим ПЗ
Інтеграції з третіх сторін (Third-Party Integrations)	Високий рівень інтеграцій знижує ймовірність появи вразливостей через зовнішні системи.
Частота резервного копіювання (Backup Frequency)	Висока частота резервного копіювання знижує бізнес-вплив від інцидентів безпеки.
Управління ключами шифрування (Encryption Key Management)	Високий рівень управління ключами знижує ймовірність компрометації шифрування.
Фізичні заходи безпеки (Physical Security Measures)	Високі фізичні заходи безпеки знижують ймовірність фізичних вторгнень та компрометації обладнання.

Також для кожного з факторів було обрано відповідні категорії, в таблиці 2 наведено категорії.

Таблиця 3 Змінні для наших категорій

Змінні для нашої моделі	Категорії
Рівень шифрування (Encryption Level)	WPA, WPA2, WPA3.
Сила паролів (Password Strength)	Слабка, Середня, Сильна.
Наявність політик безпеки (Security Policies)	Відсутні, Часткові, Повні.
Фізичне розташування точок доступу (Physical Location of Access Points)	Відкриті, Захищені.
Частота оновлень програмного забезпечення (Frequency of Software Updates)	Низька, Середня, Висока.

Рівень відповідності нормативним вимогам (Compliance Level)	Низький, Середній, Високий
Чутливість даних (Data Sensitivity)	Низька, Середня, Висока
Сегментація мережі (Network Segmentation)	Відсутня, Часткова, Повна
Системи виявлення шкідливого програмного забезпечення (Malware Detection Systems)	Низький, Середній, Високий
Інтеграції з третіх сторін (Third-Party Integrations)	Низький, Середній, Високий
Частота резервного копіювання (Backup Frequency)	Низька, Середня, Висока
Управління ключами шифрування (Encryption Key Management)	Низький, Середній, Високий
Фізичні заходи безпеки (Physical Security Measures)	Низькі, Середні, Високі

Після ідентифікації змінних необхідно визначити їхні взаємозалежності. Це включає встановлення напрямків дуг, що відображають умовні залежності між змінними, в таблиці 3 наведено всі взаємозв'язки.

Таблиця 4 Взаємозв'язки між змінними та наслідками

Наслідок	
Ймовірність несанкціонованого доступу (Unauthorized Access Probability)	- Сила паролів (Password Strength) - Наявність політик безпеки (Security Policies) - Управління ключами шифрування (Encryption Key Management)
Ймовірність перехоплення даних (Data Interception Probability)	- Рівень шифрування (Encryption Level) - Управління ключами шифрування (Encryption Key Management) - Чутливість даних (Data Sensitivity)
Ймовірність атак типу "людина посередині" (MITM Attack Probability)	- Рівень шифрування (Encryption Level) - Управління ключами шифрування (Encryption Key Management) - Чутливість даних (Data Sensitivity)
Ймовірність успішної атаки (Attack Success Probability)	- Системи виявлення шкідливого програмного забезпечення (Malware Detection Systems) - Сегментація мережі (Network Segmentation) - Частота оновлень програмного

	забезпечення (Frequency of Software Updates)
Ймовірність експлуатації вразливостей (Vulnerability Exploitation Probability)	- Частота оновлень програмного забезпечення (Frequency of Software Updates) - Системи виявлення шкідливого програмного забезпечення (Malware Detection Systems) - Управління ключами шифрування (Encryption Key Management) - Рівень відповідності нормативним вимогам (Compliance Level) - Інтеграції з третіх сторін (Third-Party Integrations)
Ймовірність фізичного вторгнення (Physical Intrusion Probability)	- Фізичне розташування точок доступу (Physical Location of Access Points) - Фізичні заходи безпеки (Physical Security Measures) - Сегментація мережі (Network Segmentation)
Загальний ризик (Overall Risk)	- Ймовірність успішної атаки (Attack Success Probability) - Ймовірність перехоплення даних (Data Interception Probability) - Ймовірність несанкціонованого доступу (Unauthorized Access Probability) - Ймовірність експлуатації вразливостей (Vulnerability Exploitation Probability) - Ймовірність фізичного вторгнення (Physical Intrusion Probability) - Інтеграції з третіх сторін (Third-Party Integrations)
Бізнес-вплив (Business Impact)	- Загальний ризик (Overall Risk) - Чутливість даних (Data Sensitivity) - Частота резервного копіювання (Backup Frequency)

3.3 Інтерпретація результатів

В результаті інтеграції цих моделей ми отримуємо результати з обох моделей, які інтерпретуємо наступним чином:

Просте Усереднення (Simple Averaging):

Просте усереднення — це найпростіший спосіб комбінування кількох ймовірнісних розподілів. Основна ідея полягає в тому, щоб для кожної категорії (наприклад, "Низький", "Середній", "Високий") обчислити середнє арифметичне ймовірностей, наданих різними моделями.

Формула:

Якщо маємо дві моделі з ймовірнісними розподілами P1 та P2, тоді об'єднана ймовірність P для кожної категорії iii обчислюється як:

$$P(i)=(P1(i)+P2(i))/2$$

Переваги

- Простота реалізації: Легко обчислити без потреби в додаткових параметрах чи налаштуваннях.
- Швидкість: Вимагає мінімальних обчислювальних ресурсів.

Недоліки

- Не враховує якість моделей: Якщо одна модель значно краща за іншу, просте усереднення не відображає цього.
- Відсутність гнучкості: Неможливо налаштувати ваги для різних моделей.

Вагове Усереднення (Weighted Averaging)

Теорія

Вагове усереднення дозволяє присвоїти різні ваги кожній моделі залежно від їхньої надійності, точності або інших критеріїв. Це означає, що деякі моделі можуть мати більший вплив на кінцевий результат, ніж інші.

Формула:

Якщо маємо nnn моделей з ймовірнісними розподілами P1,P2,…,Pn та відповідними вагами w1,w2,…,wn (де $\sum_{k=1}^{n} w_k = 1$), тоді об'єднана ймовірність P для категорії i обчислюється як:

$$P(i) = \sum_{k=1}^{n} w_k * P_i(i)$$

Переваги

- Гнучкість: Можливість відображати відмінну якість або важливість різних моделей.
- Покращення точності: Зменшує вплив менш точних моделей на кінцевий результат.

Недоліки

- Визначення ваг: Необхідно мати обґрунтовані ваги, що може вимагати додаткового аналізу або валідації.
- Складність: Вимагає додаткових параметрів для налаштування.

Множинне Комбінування (Multiplicative Combination)

Теорія

Множинне комбінування передбачає множення ймовірностей для кожної категорії, наданих різними моделями, а потім нормалізацію результатів для отримання валідного ймовірнісного розподілу (тобто, сума ймовірностей дорівнює 1).

Цей метод підкреслює узгодженість між моделями: категорія, яка має високу ймовірність у обох моделях, отримує вищу кінцеву ймовірність.

Формула

Якщо маємо nnn моделей з ймовірнісними розподілами P1,P2,…,Pn тоді об'єднана ймовірність P для категорії i обчислюється як:

$$P(i) = \sum_{j=1}^{m} \prod_{k=1}^{n} P_k(j)$$

де m — кількість категорій.

Переваги

- Підкреслює узгодженість: Висока ймовірність у всіх моделях призводить до високої кінцевої ймовірності.

- Незалежність моделей: Припускає, що моделі надають незалежні свідчення.

Недоліки

- Складність налаштування: Може бути чутливим до невеликих ймовірностей у окремих моделях.
- Не враховує ваги моделей: Якщо одна модель менш надійна, вона може негативно вплинути на кінцевий результат.

В лістингу 3.1 наведено фінальний код програмної реалізації.

Лістинг 3.1 – Програмна реалізація Баєсівської мережі

```
    def bayesian_network(test_scenario=None, seed=None):
        """
        Builds a Bayesian Network for Information Security Risk Assessment,
        visualizes it, and performs inference based on a test scenario.

        Parameters:
        - test_scenario (dict): A dictionary representing the evidence for inference.
        - seed (int): Seed for random number generation for reproducibility.

        Returns:
        - posterior_overall_risk (pgmpy.factors.discrete.DiscreteFactor): Posterior distribution of OverallRisk.
        - posterior_business_impact (pgmpy.factors.discrete.DiscreteFactor): Posterior distribution of BusinessImpact.
        """

        if seed is not None:
            np.random.seed(seed)

        def generate_random_cpd(variable_card, evidence_card):
            num_columns = np.prod(evidence_card)
            cpd_values = np.random.rand(variable_card, num_columns)
            cpd_values /= cpd_values.sum(axis=0)  # Normalize columns to sum to 1
            return cpd_values.tolist()

        # Define the structure of the Bayesian Network
        model = BayesianNetwork([
            # Dependencies for Unauthorized Access Probability
            ('PasswordStrength', 'UnauthorizedAccessProbability'),
            ('SecurityPolicies', 'UnauthorizedAccessProbability'),
```

```
            ('EncryptionKeyManagement',
'UnauthorizedAccessProbability'),

            # Dependencies for Data Interception Probability
            ('EncryptionLevel', 'DataInterceptionProbability'),
            ('EncryptionKeyManagement',
'DataInterceptionProbability'),
            ('DataSensitivity', 'DataInterceptionProbability'),

            # Dependencies for MITM Attack Probability
            ('EncryptionLevel', 'MITMAttackProbability'),
            ('EncryptionKeyManagement', 'MITMAttackProbability'),
            ('DataSensitivity', 'MITMAttackProbability'),

            # Dependencies for Attack Success Probability
            ('MalwareDetectionSystems',
'AttackSuccessProbability'),
            ('NetworkSegmentation', 'AttackSuccessProbability'),
            ('FrequencySoftwareUpdates',
'AttackSuccessProbability'),

            # Dependencies for Vulnerability Exploitation
Probability
            ('FrequencySoftwareUpdates',
'VulnerabilityExploitationProbability'),
            ('MalwareDetectionSystems',
'VulnerabilityExploitationProbability'),
            ('EncryptionKeyManagement',
'VulnerabilityExploitationProbability'),
            ('ComplianceLevel',
'VulnerabilityExploitationProbability'),
            ('ThirdPartyIntegrations',
'VulnerabilityExploitationProbability'),

            # Dependencies for Physical Intrusion Probability
            ('PhysicalLocationAccessPoints',
'PhysicalIntrusionProbability'),
            ('PhysicalSecurityMeasures',
'PhysicalIntrusionProbability'),
            ('NetworkSegmentation',
'PhysicalIntrusionProbability'),

            # Dependencies for Overall Risk
            ('AttackSuccessProbability', 'OverallRisk'),
            ('DataInterceptionProbability', 'OverallRisk'),
            ('UnauthorizedAccessProbability', 'OverallRisk'),
            ('VulnerabilityExploitationProbability',
'OverallRisk'),
            ('PhysicalIntrusionProbability', 'OverallRisk'),
            ('ThirdPartyIntegrations', 'OverallRisk'),
            ('MITMAttackProbability', 'OverallRisk'),

            # Dependencies for Business Impact
            ('OverallRisk', 'BusinessImpact'),
            ('DataSensitivity', 'BusinessImpact'),
```

```
            ('BackupFrequency', 'BusinessImpact')
        ])

        # Define CPDs for each variable

        # 1. PasswordStrength
        cpd_password_strength = TabularCPD(
            variable='PasswordStrength',
            variable_card=3,
            values=[
                [0.3],  # Weak
                [0.5],  # Medium
                [0.2]   # Strong
            ],
            state_names={'PasswordStrength':  ['Weak',  'Medium',
'Strong']}
        )

        # 2. SecurityPolicies
        cpd_security_policies = TabularCPD(
            variable='SecurityPolicies',
            variable_card=3,
            values=[
                [0.4],  # None
                [0.4],  # Partial
                [0.2]   # Full
            ],
            state_names={'SecurityPolicies':  ['None',  'Partial',
'Full']}
        )

        # 3. EncryptionKeyManagement
        cpd_encryption_key_management = TabularCPD(
            variable='EncryptionKeyManagement',
            variable_card=3,
            values=[
                [0.5],  # Low
                [0.3],  # Medium
                [0.2]   # High
            ],
            state_names={'EncryptionKeyManagement':       ['Low',
'Medium', 'High']}
        )

        # 4. UnauthorizedAccessProbability
        cpd_unauthorized_access = TabularCPD(
            variable='UnauthorizedAccessProbability',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[3, 3, 3]),
            evidence=['PasswordStrength',     'SecurityPolicies',
'EncryptionKeyManagement'],
            evidence_card=[3, 3, 3],
            state_names={
```

```
                'UnauthorizedAccessProbability': ['Low', 'Medium',
'High'],
                'PasswordStrength': ['Weak', 'Medium', 'Strong'],
                'SecurityPolicies': ['None', 'Partial', 'Full'],
                'EncryptionKeyManagement':    ['Low',    'Medium',
'High']
            }
        )

        # 5. EncryptionLevel
        cpd_encryption_level = TabularCPD(
            variable='EncryptionLevel',
            variable_card=4,
            values=[
                [0.25],  # WEP
                [0.35],  # WPA
                [0.3],   # WPA2
                [0.1]    # WPA3
            ],
            state_names={'EncryptionLevel': ['WEP', 'WPA', 'WPA2',
'WPA3']}
        )

        # 6. DataSensitivity
        cpd_data_sensitivity = TabularCPD(
            variable='DataSensitivity',
            variable_card=3,
            values=[
                [0.4],  # Low
                [0.4],  # Medium
                [0.2]   # High
            ],
            state_names={'DataSensitivity':    ['Low',    'Medium',
'High']}
        )

        # 7. DataInterceptionProbability
        cpd_data_interception = TabularCPD(
            variable='DataInterceptionProbability',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[4, 3, 3]),
            evidence=['EncryptionLevel',
'EncryptionKeyManagement', 'DataSensitivity'],
            evidence_card=[4, 3, 3],
            state_names={
                'DataInterceptionProbability':  ['Low',  'Medium',
'High'],
                'EncryptionLevel': ['WEP', 'WPA', 'WPA2', 'WPA3'],
                'EncryptionKeyManagement':    ['Low',    'Medium',
'High'],
                'DataSensitivity': ['Low', 'Medium', 'High']
            }
        )
```

```
        # 8. MITMAttackProbability
        cpd_mitm_attack = TabularCPD(
            variable='MITMAttackProbability',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[4, 3, 3]),
            evidence=['EncryptionLevel',
'EncryptionKeyManagement', 'DataSensitivity'],
            evidence_card=[4, 3, 3],
            state_names={
                'MITMAttackProbability':    ['Low',    'Medium',
'High'],
                'EncryptionLevel': ['WEP', 'WPA', 'WPA2', 'WPA3'],
                'EncryptionKeyManagement':    ['Low',    'Medium',
'High'],
                'DataSensitivity': ['Low', 'Medium', 'High']
            }
        )

        # 9. MalwareDetectionSystems
        cpd_malware_detection = TabularCPD(
            variable='MalwareDetectionSystems',
            variable_card=3,
            values=[
                [0.5],  # Low
                [0.3],  # Medium
                [0.2]   # High
            ],
            state_names={'MalwareDetectionSystems':        ['Low',
'Medium', 'High']}
        )

        # 10. NetworkSegmentation
        cpd_network_segmentation = TabularCPD(
            variable='NetworkSegmentation',
            variable_card=3,
            values=[
                [0.3],  # None
                [0.5],  # Partial
                [0.2]   # Full
            ],
            state_names={'NetworkSegmentation': ['Low', 'Medium',
'High']}
        )

        # 11. FrequencySoftwareUpdates
        cpd_frequency_software_updates = TabularCPD(
            variable='FrequencySoftwareUpdates',
            variable_card=3,
            values=[
                [0.4],  # Low
                [0.4],  # Medium
                [0.2]   # High
            ],
```

```
            state_names={'FrequencySoftwareUpdates':       ['Low',
'Medium', 'High']}
        )

        # 12. AttackSuccessProbability
        cpd_attack_success = TabularCPD(
            variable='AttackSuccessProbability',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[3, 3, 3]),
            evidence=['MalwareDetectionSystems',
'NetworkSegmentation', 'FrequencySoftwareUpdates'],
            evidence_card=[3, 3, 3],
            state_names={
                'AttackSuccessProbability':    ['Low',    'Medium',
'High'],
                'MalwareDetectionSystems':    ['Low',    'Medium',
'High'],
                'NetworkSegmentation': ['Low', 'Medium', 'High'],
                'FrequencySoftwareUpdates':    ['Low',    'Medium',
'High']
            }
        )

        # 13. VulnerabilityExploitationProbability
        cpd_vulnerability_exploitation = TabularCPD(
            variable='VulnerabilityExploitationProbability',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[3, 3, 3, 3, 3]),
            evidence=['FrequencySoftwareUpdates',
'MalwareDetectionSystems',               'EncryptionKeyManagement',
'ComplianceLevel', 'ThirdPartyIntegrations'],
            evidence_card=[3, 3, 3, 3, 3],
            state_names={
                'VulnerabilityExploitationProbability':    ['Low',
'Medium', 'High'],
                'FrequencySoftwareUpdates':    ['Low',    'Medium',
'High'],
                'MalwareDetectionSystems':    ['Low',    'Medium',
'High'],
                'EncryptionKeyManagement':    ['Low',    'Medium',
'High'],
                'ComplianceLevel': ['Low', 'Medium', 'High'],
                'ThirdPartyIntegrations':    ['Low',    'Medium',
'High']
            }
        )

        # 14. PhysicalLocationAccessPoints
        cpd_physical_location = TabularCPD(
            variable='PhysicalLocationAccessPoints',
            variable_card=2,
            values=[
                [0.6],  # Open
```

```
                [0.4]   # Protected
            ],
            state_names={'PhysicalLocationAccessPoints':  ['Open',
'Protected']}
        )

        # 15. PhysicalSecurityMeasures
        cpd_physical_security_measures = TabularCPD(
            variable='PhysicalSecurityMeasures',
            variable_card=3,
            values=[
                [0.5],  # Low
                [0.3],  # Medium
                [0.2]   # High
            ],
            state_names={'PhysicalSecurityMeasures':       ['Low',
'Medium', 'High']}
        )

        # 16. PhysicalIntrusionProbability
        cpd_physical_intrusion = TabularCPD(
            variable='PhysicalIntrusionProbability',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[2, 3, 3]),
            evidence=['PhysicalLocationAccessPoints',
'PhysicalSecurityMeasures', 'NetworkSegmentation'],
            evidence_card=[2, 3, 3],
            state_names={
                'PhysicalIntrusionProbability': ['Low',  'Medium',
'High'],
                'PhysicalLocationAccessPoints':           ['Open',
'Protected'],
                'PhysicalSecurityMeasures':   ['Low',   'Medium',
'High'],
                'NetworkSegmentation': ['Low', 'Medium', 'High']
            }
        )

        # 17. OverallRisk
        cpd_overall_risk = TabularCPD(
            variable='OverallRisk',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[3, 3, 3, 3, 3, 3, 3]),
            evidence=['AttackSuccessProbability',
'DataInterceptionProbability', 'UnauthorizedAccessProbability',
                      'VulnerabilityExploitationProbability',
'PhysicalIntrusionProbability',

'ThirdPartyIntegrations','MITMAttackProbability'],
            evidence_card=[3, 3, 3, 3, 3, 3, 3],
            state_names={
                'OverallRisk': ['Low', 'Medium', 'High'],
```

```
                'AttackSuccessProbability':   ['Low',   'Medium',
'High'],
                'DataInterceptionProbability':  ['Low',  'Medium',
'High'],
                'UnauthorizedAccessProbability': ['Low', 'Medium',
'High'],
                'VulnerabilityExploitationProbability':   ['Low',
'Medium', 'High'],
                'PhysicalIntrusionProbability': ['Low',  'Medium',
'High'],
                'MITMAttackProbability':    ['Low',    'Medium',
'High'],
                'ThirdPartyIntegrations':    ['Low',    'Medium',
'High']
            }
        )

        # 18. ThirdPartyIntegrations
        cpd_third_party_integrations = TabularCPD(
            variable='ThirdPartyIntegrations',
            variable_card=3,
            values=[
                [0.5],  # Low
                [0.3],  # Medium
                [0.2]   # High
            ],
            state_names={'ThirdPartyIntegrations':         ['Low',
'Medium', 'High']}
        )

        # 19. ComplianceLevel
        cpd_compliance_level = TabularCPD(
            variable='ComplianceLevel',
            variable_card=3,
            values=[
                [0.3],  # Low
                [0.5],  # Medium
                [0.2]   # High
            ],
            state_names={'ComplianceLevel':   ['Low',   'Medium',
'High']}
        )

        # 20. BackupFrequency
        cpd_backup_frequency = TabularCPD(
            variable='BackupFrequency',
            variable_card=3,
            values=[
                [0.4],  # Low
                [0.4],  # Medium
                [0.2]   # High
            ],
            state_names={'BackupFrequency':   ['Low',   'Medium',
'High']}
        )
```

```
        # 21. BusinessImpact
        cpd_business_impact = TabularCPD(
            variable='BusinessImpact',
            variable_card=3,
            values=generate_random_cpd(variable_card=3,
evidence_card=[3, 3, 3]),
            evidence=['OverallRisk',           'DataSensitivity',
'BackupFrequency'],
            evidence_card=[3, 3, 3],
            state_names={
                'BusinessImpact': ['Low', 'Medium', 'High'],
                'OverallRisk': ['Low', 'Medium', 'High'],
                'DataSensitivity': ['Low', 'Medium', 'High'],
                'BackupFrequency': ['Low', 'Medium', 'High']
            }
        )

        # Add all CPDs to the model
        model.add_cpds(
            cpd_password_strength,
            cpd_security_policies,
            cpd_encryption_key_management,
            cpd_unauthorized_access,
            cpd_encryption_level,
            cpd_data_sensitivity,
            cpd_data_interception,
            cpd_mitm_attack,
            cpd_malware_detection,
            cpd_network_segmentation,
            cpd_frequency_software_updates,
            cpd_attack_success,
            cpd_vulnerability_exploitation,
            cpd_physical_location,
            cpd_physical_security_measures,
            cpd_physical_intrusion,
            cpd_overall_risk,
            cpd_backup_frequency,
            cpd_business_impact,
            cpd_third_party_integrations,
            cpd_compliance_level
        )

        # Check the model for correctness
        try:
            model.check_model()
            print("Модель побудована коректно.")
        except ValueError as e:
            print(f"Модель містить помилки: {e}")
            return None, None

        # Define categories for nodes
        risk_factors = [
            'PasswordStrength',               'SecurityPolicies',
'EncryptionKeyManagement',
```

```
            'EncryptionLevel',                  'DataSensitivity',
'MalwareDetectionSystems',
            'NetworkSegmentation', 'FrequencySoftwareUpdates',
            'PhysicalLocationAccessPoints',
'PhysicalSecurityMeasures',
            'ComplianceLevel',          'ThirdPartyIntegrations',
'BackupFrequency'
        ]

        risk_probabilities = [
            'UnauthorizedAccessProbability',
'DataInterceptionProbability',
            'MITMAttackProbability', 'AttackSuccessProbability',
            'VulnerabilityExploitationProbability',
'PhysicalIntrusionProbability'
        ]

        consequences = [
            'OverallRisk', 'BusinessImpact'
        ]

        # Assign colors to nodes based on categories
        color_map = {}
        layer_map = {}
        for node in model.nodes():
            if node in risk_factors:
                color_map[node] = 'skyblue'  # Blue for Risks
                layer_map[node] = 1  # Top layer
            elif node in risk_probabilities:
                color_map[node] = 'gold'       # Yellow for Risk
Probabilities
                layer_map[node] = 2  # Middle layer
            elif node in consequences:
                color_map[node]   =   'lightcoral'     #   Red   for
Consequences
                layer_map[node] = 3  # Bottom layer
            else:
                color_map[node] = 'grey'      # Grey for others
                layer_map[node] = 2  # Default to middle layer

        # Create the graph for visualization
        graph_nx = nx.DiGraph()
        graph_nx.add_edges_from(model.edges())

        # Define positions manually to layer nodes
        pos = {}
        layer_nodes = {}
        for node, layer in layer_map.items():
            if layer not in layer_nodes:
                layer_nodes[layer] = []
            layer_nodes[layer].append(node)

        # Assign vertical positions for each layer
        y_positions = {1: 2, 2: 1, 3: 0}  # Adjust these values to
change spacing between layers
```

```
        # Arrange nodes in each layer horizontally
        for layer in sorted(layer_nodes.keys()):
            nodes_in_layer = layer_nodes[layer]
            if len(nodes_in_layer) == 1:
                pos[nodes_in_layer[0]] = (0.5, y_positions[layer])
            else:
                x_spacing = 1.0 / (len(nodes_in_layer) + 1)
                for i, node in enumerate(nodes_in_layer, start=1):
                    pos[node] = (i * x_spacing, y_positions[layer])

        # Visualize the Bayesian Network
        plt.figure(figsize=(25, 20))
        nx.draw(
            graph_nx, pos, with_labels=True,
            node_color=[color_map[node]      for      node      in
graph_nx.nodes()],
            node_size=3000,     arrows=True,     arrowstyle='->',
arrowsize=20,
            font_size=12, font_weight='bold'
        )

        # Add legend
        legend_patches = [
        mpatches.Patch(color='skyblue', label='Ризики'),
        mpatches.Patch(color='gold', label='Ймовірності Ризиків'),
        mpatches.Patch(color='lightcoral', label='Наслідки')
    ]
        plt.legend(handles=legend_patches,   loc='upper   right',
fontsize=15)

        plt.title('Байєсівська   Мережа   для   Оцінки   Ризиків
Інформаційної Безпеки', fontsize=20)
        plt.axis('off')
        plt.show()

        # Perform inference using Variable Elimination
        infer = VariableElimination(model)

        # Define a default test scenario if none is provided
        if test_scenario is None:
            test_scenario = {
                'PasswordStrength': 'Weak',
                'SecurityPolicies': 'Full',
                'EncryptionKeyManagement': 'High',
                'EncryptionLevel': 'WPA3',
                'DataSensitivity': 'High',
                'MalwareDetectionSystems': 'High',
                'NetworkSegmentation': 'Full',
                'FrequencySoftwareUpdates': 'High',
                'PhysicalLocationAccessPoints': 'Protected',
                'PhysicalSecurityMeasures': 'High',
                'ComplianceLevel': 'High',
                'ThirdPartyIntegrations': 'Low'
            }
```

```
        # Perform inference for OverallRisk
        posterior_overall_risk = infer.query(
            variables=['OverallRisk'],
            evidence=test_scenario
        )

        print("\nOverall Risk Probability Distribution:")
        print(posterior_overall_risk)

        # Extract probabilities and states
        overall_risk_values = posterior_overall_risk.values
        overall_risk_states =
posterior_overall_risk.state_names['OverallRisk']

        # Determine the most probable state of OverallRisk
        most_probable_overall_risk_index =
overall_risk_values.argmax()
        most_probable_overall_risk =
overall_risk_states[most_probable_overall_risk_index]

        print(f"\nMost Probable Overall Risk:
{most_probable_overall_risk}")

        # Perform inference for BusinessImpact based on OverallRisk
        posterior_business_impact = infer.query(
            variables=['BusinessImpact'],
            evidence={'OverallRisk': most_probable_overall_risk}
        )

        print("\nBusiness Impact Probability Distribution:")
        print(posterior_business_impact)


        overall_risk_values = posterior_overall_risk.values
        overall_risk_states =
posterior_overall_risk.state_names['OverallRisk']

      # Розраховуємо суму всіх ймовірностей
        total_sum = sum(overall_risk_values)

      # Нормалізуємо кожне значення до інтервалу 0-1
        normalized_overall_risk = {state: prob / total_sum for
state, prob in zip(overall_risk_states, overall_risk_values)}

        business_impact_values = posterior_business_impact.values
        business_impact_states =
posterior_business_impact.state_names['BusinessImpact']

      # Розраховуємо суму всіх ймовірностей
        total_sum = sum(business_impact_values)

      # Нормалізуємо кожне значення до інтервалу 0-1
        normalized_business_impact = {state: prob / total_sum for
state, prob in zip(business_impact_states, business_impact_values)}
```

```
return normalized_overall_risk, normalized_business_impact
```

Опис функціоналу - Функція bayesian_network призначена для побудови Байєсівської мережі, яка використовується для оцінки ризиків інформаційної безпеки. Вона виконує наступні основні етапи:

Параметри функції:

- **test_scenario (dict, необов'язково):** Словник, що містить дані про наявні докази (елементи), на основі яких буде виконуватися інференція (виведення висновків).
- **seed (int, необов'язково):** Цілий номер для ініціалізації генератора випадкових чисел, що забезпечує відтворюваність результатів.

Повертає:

- **posterior_overall_risk (pgmpy.factors.discrete.DiscreteFactor):** Постеріорний розподіл ймовірності для змінної OverallRisk.
- **posterior_business_impact (pgmpy.factors.discrete.DiscreteFactor):** Постеріорний розподіл ймовірності для змінної BusinessImpact.

Опис роботи функції:

1. **Ініціалізація генератора випадкових чисел:** Якщо задано параметр seed, встановлюється відповідне значення для генератора випадкових чисел, що дозволяє отримувати однакові результати при повторних викликах функції.
2. **Визначення структури Байєсівської мережі:** Створюється об'єкт BayesianNetwork з заданими залежностями між різними змінними. Залежності відображають, як одна змінна впливає на іншу у контексті оцінки ризиків інформаційної безпеки.
3. **Визначення ймовірнісних розподілів (CPDs):** Для кожної змінної у мережі визначаються її умовні ймовірнісні розподіли. Для деяких змінних використовуються фіксовані ймовірності, а для інших — випадкові, згенеровані функцією generate_random_cpd. Це дозволяє моделювати складні залежності між змінними.

4. **Додавання CPDs до моделі:** Всі визначені CPD додаються до Байєсівської мережі.

5. **Перевірка коректності моделі:** Використовується метод check_model, щоб впевнитися, що мережа побудована правильно і всі ймовірнісні розподіли узгоджені.

6. **Візуалізація Байєсівської мережі:**

- **Категоризація вузлів:** Вузли мережі поділяються на категорії: ризики, ймовірності ризиків та наслідки. Кожній категорії призначається свій колір для візуального розрізнення.
- **Розташування вузлів:** Визначаються координати вузлів для побудови графіка з урахуванням шарів (ризики на верхньому шарі, ймовірності ризиків у середньому, наслідки в нижньому).
- **Побудова графіку:** Використовуючи бібліотеку networkx та matplotlib, створюється графічне зображення мережі з відповідними кольорами та легендою.

7. **Інференція (виведення висновків):**

- **Ініціалізація інференційного механізму:** Створюється об'єкт VariableElimination для виконання інференції в мережі.
- **Визначення тестового сценарію:** Якщо test_scenario не заданий, використовується стандартний сценарій з попередньо визначеними значеннями для певних змінних.
- **Обчислення постеріорного розподілу для OverallRisk:** Виконується запит до мережі з урахуванням наявних доказів, отримуючи ймовірнісний розподіл для загального ризику.
- **Визначення найбільш ймовірного стану OverallRisk:** Аналізується отриманий розподіл і вибирається стан з найбільшою ймовірністю.
- **Обчислення постеріорного розподілу для BusinessImpact:** На основі найбільш ймовірного значення OverallRisk виконується додатковий запит для оцінки впливу на бізнес.

8. **Нормалізація ймовірностей:** Отримані постеріорні розподіли для OverallRisk та BusinessImpact нормалізуються, щоб сума ймовірностей кожного

розподілу дорівнювала 1. Це забезпечує коректне відображення ймовірностей у межах інтервалу [0, 1].

Лістинг 3.2 – Програмна реалізація Марковських мереж

```
   def markov_network(test_scenario=None):

     state_names = {
         'PasswordStrength': ['Weak', 'Moderate', 'Strong'],
         'SecurityPolicies': ['None', 'Partial', 'Full'],
         'EncryptionKeyManagement': ['Low', 'Medium', 'High'],
         'UnauthorizedAccessProbability':     ['Low',     'Medium',
'High'],
         'EncryptionLevel': ['WPA', 'WPA2', 'WPA3'],
         'DataSensitivity': ['Low', 'Medium', 'High'],
         'DataInterceptionProbability': ['Low', 'Medium', 'High'],
         'MITMAttackProbability': ['Low', 'Medium', 'High'],
         'MalwareDetectionSystems': ['Low', 'Medium', 'High'],
         'NetworkSegmentation': ['Low', 'Medium', 'High'],
         'FrequencySoftwareUpdates': ['Low', 'Medium', 'High'],
         'AttackSuccessProbability': ['Low', 'Medium', 'High'],
         'ThirdPartyIntegrations': ['Low', 'Medium', 'High'],
         'ComplianceLevel': ['Low', 'Medium', 'High'],
         'VulnerabilityExploitationProbability': ['Low', 'Medium',
'High'],
         'PhysicalLocationAccessPoints': ['Unprotected', 'Partial',
'Protected'],
         'PhysicalSecurityMeasures': ['Low', 'Medium', 'High'],
         'PhysicalIntrusionProbability': ['Low', 'Medium', 'High'],
         'OverallRisk': ['Low', 'Medium', 'High'],
         'BusinessImpact': ['Low', 'Medium', 'High'],
         'BackupFrequency': ['Rare', 'Occasional', 'Frequent']
     }


     def complete_test_scenario(test_scenario, state_names):


       #
       completed_scenario = test_scenario.copy()

       #
       for key, states in state_names.items():
           if key not in completed_scenario:
               #
               if states:
                   #
                   completed_scenario[key] = states[0]
                   print(f"  ключ  '{key}'  доданий  зі  значенням
'{states[0]}'.")
               else:
                   #
```

```
                    completed_scenario[key] = None
                    print(f"ключ '{key}' доданий зі значенням None
.")

        return completed_scenario

    test_scenario=complete_test_scenario(test_scenario,
state_names)

    # Створення Марковської мережі
    model = MarkovModel()

    # Перелік змінних
    variables = [
        'PasswordStrength',
        'SecurityPolicies',
        'EncryptionKeyManagement',
        'UnauthorizedAccessProbability',
        'EncryptionLevel',
        'DataSensitivity',
        'DataInterceptionProbability',
        'MITMAttackProbability',
        'MalwareDetectionSystems',
        'NetworkSegmentation',
        'FrequencySoftwareUpdates',
        'AttackSuccessProbability',
        'ThirdPartyIntegrations',
        'ComplianceLevel',
        'VulnerabilityExploitationProbability',
        'PhysicalLocationAccessPoints',
        'PhysicalSecurityMeasures',
        'PhysicalIntrusionProbability',
        'OverallRisk',
        'BusinessImpact',
        'BackupFrequency'
    ]

    # Додавання вузлів (змінних)
    model.add_nodes_from(variables)

    # Додавання ребер (ненаправлених зв'язків)
    edges = [
        # Ненаправлені ребра з Байєсівської мережі
        ('PasswordStrength', 'UnauthorizedAccessProbability'),
        ('SecurityPolicies', 'UnauthorizedAccessProbability'),
        ('EncryptionKeyManagement',
'UnauthorizedAccessProbability'),

        # Ребра між батьками (моралізація)
        ('PasswordStrength', 'SecurityPolicies'),
        ('PasswordStrength', 'EncryptionKeyManagement'),
        ('SecurityPolicies', 'EncryptionKeyManagement'),

        # Залежності для DataInterceptionProbability
        ('DataSensitivity', 'DataInterceptionProbability'),
```

```
        ('EncryptionLevel', 'DataInterceptionProbability'),
        ('EncryptionKeyManagement',
'DataInterceptionProbability'),

        # Залежності для MITMAttackProbability
        ('DataSensitivity', 'MITMAttackProbability'),
        ('EncryptionLevel', 'MITMAttackProbability'),
        ('EncryptionKeyManagement', 'MITMAttackProbability'),

        # Додаткові зв'язки між змінними
        ('DataSensitivity', 'EncryptionLevel'),
        ('DataSensitivity', 'EncryptionKeyManagement'),
        ('EncryptionLevel', 'EncryptionKeyManagement'),

        # Залежності для AttackSuccessProbability
        ('MalwareDetectionSystems', 'AttackSuccessProbability'),
        ('NetworkSegmentation', 'AttackSuccessProbability'),
        ('FrequencySoftwareUpdates', 'AttackSuccessProbability'),

        # Ребра між батьками для AttackSuccessProbability
        ('MalwareDetectionSystems', 'NetworkSegmentation'),
        ('MalwareDetectionSystems', 'FrequencySoftwareUpdates'),
        ('NetworkSegmentation', 'FrequencySoftwareUpdates'),

        # Залежності для VulnerabilityExploitationProbability
        ('MalwareDetectionSystems',
'VulnerabilityExploitationProbability'),
        ('EncryptionKeyManagement',
'VulnerabilityExploitationProbability'),
        ('FrequencySoftwareUpdates',
'VulnerabilityExploitationProbability'),
        ('ThirdPartyIntegrations',
'VulnerabilityExploitationProbability'),
        ('ComplianceLevel',
'VulnerabilityExploitationProbability'),

        #             Додаткові             зв'язки             для
VulnerabilityExploitationProbability
        ('MalwareDetectionSystems', 'EncryptionKeyManagement'),
        ('MalwareDetectionSystems', 'ThirdPartyIntegrations'),
        ('MalwareDetectionSystems', 'ComplianceLevel'),
        ('EncryptionKeyManagement', 'ThirdPartyIntegrations'),
        ('EncryptionKeyManagement', 'ComplianceLevel'),
        ('FrequencySoftwareUpdates', 'ThirdPartyIntegrations'),
        ('FrequencySoftwareUpdates', 'ComplianceLevel'),
        ('ThirdPartyIntegrations', 'ComplianceLevel'),

        # Залежності для PhysicalIntrusionProbability
        ('NetworkSegmentation', 'PhysicalIntrusionProbability'),
        ('PhysicalLocationAccessPoints',
'PhysicalIntrusionProbability'),
        ('PhysicalSecurityMeasures',
'PhysicalIntrusionProbability'),

        # Ребра між батьками для PhysicalIntrusionProbability
```

```
        ('NetworkSegmentation', 'PhysicalLocationAccessPoints'),
        ('NetworkSegmentation', 'PhysicalSecurityMeasures'),
        ('PhysicalSecurityMeasures',
'PhysicalLocationAccessPoints'),

        # Залежності для OverallRisk
        ('DataInterceptionProbability', 'OverallRisk'),
        ('UnauthorizedAccessProbability', 'OverallRisk'),
        ('MITMAttackProbability', 'OverallRisk'),
        ('AttackSuccessProbability', 'OverallRisk'),
        ('VulnerabilityExploitationProbability', 'OverallRisk'),
        ('PhysicalIntrusionProbability', 'OverallRisk'),

        # Додаткові зв'язки між змінними, що впливають на
OverallRisk
        ('DataInterceptionProbability',
'UnauthorizedAccessProbability'),
        ('DataInterceptionProbability', 'MITMAttackProbability'),
        ('DataInterceptionProbability',
'AttackSuccessProbability'),
        ('DataInterceptionProbability',
'VulnerabilityExploitationProbability'),
        ('DataInterceptionProbability',
'PhysicalIntrusionProbability'),
        ('UnauthorizedAccessProbability',
'MITMAttackProbability'),
        ('UnauthorizedAccessProbability',
'AttackSuccessProbability'),
        ('UnauthorizedAccessProbability',
'VulnerabilityExploitationProbability'),
        ('UnauthorizedAccessProbability',
'PhysicalIntrusionProbability'),
        ('MITMAttackProbability', 'AttackSuccessProbability'),
        ('MITMAttackProbability',
'VulnerabilityExploitationProbability'),
        ('MITMAttackProbability', 'PhysicalIntrusionProbability'),
        ('AttackSuccessProbability',
'VulnerabilityExploitationProbability'),
        ('AttackSuccessProbability',
'PhysicalIntrusionProbability'),
        ('VulnerabilityExploitationProbability',
'PhysicalIntrusionProbability'),

        # Залежності для BusinessImpact
        ('OverallRisk','BusinessImpact'),
        ('DataSensitivity','BusinessImpact'),
        ('BackupFrequency','BusinessImpact'),

        # Додаткові зв'язки для BusinessImpact
        ('OverallRisk','DataSensitivity'),
        ('OverallRisk','BackupFrequency'),
        ('BackupFrequency','DataSensitivity'),
    ]

    model.add_edges_from(edges)
```

```
    # Визначення кардинальності для кожної змінної
    cardinality = {var: len(state_names[var]) for var in variables}

    # Визначення кореневих (root) змінних (без впливових змінних)
    non_root_variables = [
        'UnauthorizedAccessProbability',
        'DataInterceptionProbability',
        'MITMAttackProbability',
        'AttackSuccessProbability',
        'VulnerabilityExploitationProbability',
        'PhysicalIntrusionProbability',
        'OverallRisk',
        'BusinessImpact'
    ]

    root_variables = [var for var in variables if var not in
non_root_variables]

    # Функція для створення та додавання факторів для початкових
змінних
    def   create_initial_factor(model,   variable,   state_names,
cardinality, potential_values=None):
        """
        Створює та додає фактор до моделі для початкової змінної
(без впливових змінних).

        :param model: Марковська модель (MarkovModel).
        :param variable: Назва змінної (str).
        :param state_names: Словник зі станами змінних (dict).
        :param  cardinality:  Словник  з  кардинальністю  змінних
(dict).
        :param potential_values: Список або масив ймовірностей для
станів змінної (list або np.array).
                                 Якщо    None,    використовується
рівномірний розподіл.
        """
        variables = [variable]
        card = [cardinality[variable]]

        if potential_values is None:
            # Рівномірний розподіл
            potential_values = np.ones(cardinality[variable])
            # Не нормалізуємо потенціали в Марковських мережах
        else:
            potential_values     =     np.array(potential_values,
dtype=float)
            # Перевірка сумарної ймовірності (не обов'язково для
Марковських мереж)
            # if not np.isclose(np.sum(potential_values), 1.0):
            #     potential_values /= np.sum(potential_values)

        # Створення фактора
        factor = DiscreteFactor(
            variables,
```

```
            card,
            potential_values,
            state_names={variable: state_names[variable]}
        )

        # Додавання фактора до моделі
        model.add_factors(factor)
        print(f"Фактор для {variables} додано.\n")

    # Створення факторів для кореневих змінних
    for var in root_variables:
        potential = None
        if var == 'PasswordStrength':
            potential = [0.3, 0.5, 0.2]  # Weak, Moderate, Strong
        elif var == 'SecurityPolicies':
            potential = [0.2, 0.6, 0.2]  # None, Partial, Full
        elif var == 'EncryptionKeyManagement':
            potential = [0.4, 0.5, 0.1]  # Low, Medium, High
        elif var == 'EncryptionLevel':
            potential = [0.3, 0.6, 0.1]  # WPA, WPA2, WPA3
        elif var == 'DataSensitivity':
            potential = [0.5, 0.3, 0.2]  # Low, Medium, High
        elif       var       in       ['MalwareDetectionSystems',
'NetworkSegmentation', 'FrequencySoftwareUpdates',
                     'ComplianceLevel',   'ThirdPartyIntegrations',
'PhysicalLocationAccessPoints',
                     'PhysicalSecurityMeasures',
'BackupFrequency']:
            potential = [0.5, 0.3, 0.2]  # Low, Medium, High або
відповідно
        else:
            potential = None  # Для додаткової безпеки

        create_initial_factor(
            model=model,
            variable=var,
            state_names=state_names,
            cardinality=cardinality,
            potential_values=potential
        )

    # Функція для створення та додавання факторів для змінних з
впливовими змінними
    def create_and_add_factor(model, target_var, influencer_vars,
state_names, cardinality):
        """
        Створює та додає фактор до моделі для заданої цільової
змінної та її впливових змінних.

        :param model: Марковська модель
        :param target_var: Цільова змінна (str)
        :param influencer_vars: Список впливових змінних (list of
str)
        :param state_names: Словник зі станами змінних (dict)
        :param cardinality: Словник з кардинальністю змінних (dict)
```

```
        """
        variables = [target_var] + influencer_vars
        card = [cardinality[var] for var in variables]

        for var1 in variables:
            for var2 in variables:
                if var1 != var2 and (var1, var2) not in model.edges
and (var2, var1) not in model.edges:
                    model.add_edge(var1, var2)
                    print(f"Додано ребро: ({var1}, {var2})")


        # Генерація всіх можливих комбінацій станів
        combinations = list(product(*[range(c) for c in card]))

        # Ініціалізація потенціалу
        data_for_potential = np.zeros(len(combinations))

        for idx, combo in enumerate(combinations):
            # combo[0] - target_var state
            # combo[1:] - influencing_vars states
            target_state = combo[0]
            influencing_states = combo[1:]

            # Логіка потенціалу:
            # Якщо більше половини впливових змінних в стані
'High', target_var в стані 'High'
            # Якщо більше половини впливових змінних в стані 'Low',
target_var в стані 'Low'
            # Інакше target_var в стані 'Medium'

            high_count = sum(state == (cardinality[var]-1) for var,
state in zip(influencer_vars, influencing_states))
            low_count = sum(state == 0 for state in
influencing_states)
            total = len(influencer_vars)

            if high_count > total / 2 and target_state ==
(cardinality[target_var]-1):
                potential = 10.0  # Високий потенціал для 'High'
            elif low_count > total / 2 and target_state == 0:
                potential = 10.0  # Високий потенціал для 'Low'
            elif target_state == 1:
                potential = 5.0   # Середній потенціал для 'Medium'
            else:
                potential = 1.0    # Низький потенціал для інших
випадків

            data_for_potential[idx] = potential

        # Не нормалізуємо потенціали в Марковських мережах
        # data_for_potential /= np.sum(data_for_potential)

        # Створення фактора
        factor = DiscreteFactor(
```

```
            variables,
            card,
            data_for_potential,
            state_names={var:   state_names[var]   for   var   in
variables}
        )

        # Додавання фактора до моделі
        model.add_factors(factor)
        print(f"Фактор для {variables} додано.\n")

    # Створення факторів для змінних з впливовими змінними
    create_and_add_factor(
        model=model,
        target_var='UnauthorizedAccessProbability',
        influencer_vars=['PasswordStrength',   'SecurityPolicies',
'EncryptionKeyManagement'],
        state_names=state_names,
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='DataInterceptionProbability',
        influencer_vars=['EncryptionLevel',
'EncryptionKeyManagement', 'DataSensitivity'],
        state_names=state_names,
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='MITMAttackProbability',
        influencer_vars=['EncryptionLevel',
'EncryptionKeyManagement', 'DataSensitivity'],
        state_names=state_names,
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='AttackSuccessProbability',
        influencer_vars=['MalwareDetectionSystems',
'NetworkSegmentation', 'FrequencySoftwareUpdates'],
        state_names=state_names,
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='VulnerabilityExploitationProbability',
        influencer_vars=['FrequencySoftwareUpdates',
'MalwareDetectionSystems',              'EncryptionKeyManagement',
'ThirdPartyIntegrations', 'ComplianceLevel'],
        state_names=state_names,
```

```
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='PhysicalIntrusionProbability',
influencer_vars=['NetworkSegmentation','PhysicalLocationAccessPoin
ts','PhysicalSecurityMeasures'],
        state_names=state_names,
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='OverallRisk',
        influencer_vars=[
            'DataInterceptionProbability',
            'UnauthorizedAccessProbability',
            'MITMAttackProbability',
            'AttackSuccessProbability',
            'VulnerabilityExploitationProbability',
            'PhysicalIntrusionProbability'
        ],
        state_names=state_names,
        cardinality=cardinality
    )

    create_and_add_factor(
        model=model,
        target_var='BusinessImpact',
        influencer_vars=['OverallRisk',         'DataSensitivity',
'BackupFrequency'],
        state_names=state_names,
        cardinality=cardinality
    )

    # Перевірка, чи всі фактори створені для кореневих змінних
    # Оскільки кореневі змінні вже мають фактори, немає потреби
додавати додаткові фактори
    # Це уникає дублювання факторів для змінних з впливовими
змінними

    # Додатково, можна переконатися, що немає факторів для
non_root_variables
    for var in non_root_variables:
        if model.get_factors(var):
            print(f"Попередження: Фактор для {var} вже існує як
частина іншого фактора.")

    # Перевірка моделі
    if model.check_model():
        print("Модель коректна.")
    else:
```

```
        print("Модель некоректна. Перевірте фактори та їхні
потенціали.")

    # Створення об'єкта інференції



    from pgmpy.inference import VariableElimination

    # Створення об'єкта інференції
    infer = VariableElimination(model)

    #infer = BeliefPropagation(model)

    # Встановлення доказів (Evidence)
    evidence = {

    }

    # Виконання запиту
    #result                                                    =
infer.query(variables=['DataSensitivity','OverallRisk','BackupFreq
uency','BusinessImpact'], show_progress=True)
    #map_result                                                =
infer.map_query(variables=['DataSensitivity','OverallRisk','Backup
Frequency','BusinessImpact'], evidence=evidence)

    # Виведення результатів
    #print("Найбільш ймовірна комбінація станів:")
    #for var, state in map_result.items():
    #    print(f"{var}: {state}")

    UAP_result                                                =
infer.query(variables=['UnauthorizedAccessProbability'],
evidence={'PasswordStrength':    test_scenario['PasswordStrength'],
'SecurityPolicies':              test_scenario['SecurityPolicies'],
'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement']}, show_progress=True)
    DIP_result                                                =
infer.query(variables=['DataInterceptionProbability'],
evidence={'EncryptionLevel':      test_scenario['EncryptionLevel'],
'DataSensitivity':                test_scenario['DataSensitivity'],
'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement']}, show_progress=True)
    MitmAP_result                                             =
infer.query(variables=['MITMAttackProbability'],
evidence={'PasswordStrength':    test_scenario['PasswordStrength'],
'SecurityPolicies':              test_scenario['SecurityPolicies'],
'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement']}, show_progress=True)
    ASP_result                                                =
infer.query(variables=['AttackSuccessProbability'],
evidence={'MalwareDetectionSystems':
test_scenario['MalwareDetectionSystems'],    'NetworkSegmentation':
```

```
test_scenario['NetworkSegmentation'],   'FrequencySoftwareUpdates':
test_scenario['FrequencySoftwareUpdates']}, show_progress=True)
     VEP_result                                                    =
infer.query(variables=['VulnerabilityExploitationProbability'],
evidence={'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement'],
'MalwareDetectionSystems':
test_scenario['MalwareDetectionSystems'],
'FrequencySoftwareUpdates':
test_scenario['FrequencySoftwareUpdates'],      'ComplianceLevel':
test_scenario['ComplianceLevel'],'ThirdPartyIntegrations':
test_scenario['ThirdPartyIntegrations']}, show_progress=True)
     PIP_result                                                    =
infer.query(variables=['PhysicalIntrusionProbability'],
evidence={'NetworkSegmentation':
test_scenario['NetworkSegmentation'],
'PhysicalLocationAccessPoints':
test_scenario['PhysicalLocationAccessPoints'],
'PhysicalSecurityMeasures':
test_scenario['PhysicalSecurityMeasures']}, show_progress=True)


     #M2_result  =  infer.query(variables=['M2'],  evidence={'B4':
'value4', 'B5': 'value5', 'B6': 'value6'}, show_progress=True)
     # Повторіть для кожної середньої змінної M3, M4, ..., M6

     # Зберігаємо результати часткових симуляцій середніх змінних в
словнику для подальшого використання
     def get_most_probable_state(result):
         max_index  =  result.values.argmax()    #  Індекс  стану  з
максимальною ймовірністю
         return  result.state_names[result.variables[0]][max_index]
# Назва стану

     partial_results = {
         'UAP': get_most_probable_state(UAP_result),
         'DIP': get_most_probable_state(DIP_result),
         'MitmAP': get_most_probable_state(MitmAP_result),
         'ASP': get_most_probable_state(ASP_result),
         'VEP': get_most_probable_state(VEP_result),
         'PIP': get_most_probable_state(PIP_result),
     }
     #print(partial_results)

     OR_result       =       infer.query(variables=['OverallRisk'],
evidence={'UnauthorizedAccessProbability':  partial_results['UAP'],
'DataInterceptionProbability': partial_results['DIP'],

'MITMAttackProbability': partial_results['MitmAP'] ,

'AttackSuccessProbability': partial_results['ASP'],

'VulnerabilityExploitationProbability': partial_results['VEP'],

'PhysicalIntrusionProbability': partial_results['PIP']
```

```
}, show_progress=True)

    #print(OR_result)
    business_impact_result = infer.query(
        variables=['BusinessImpact'],
        evidence={'OverallRisk':
get_most_probable_state(OR_result),              'DataSensitivity':
test_scenario['DataSensitivity'],                'BackupFrequency':
test_scenario['BackupFrequency']},
        show_progress=True
    )

    print("Overall Risk:", OR_result)
    print("Business Impact:", business_impact_result)
    print("Overall Risk:", get_most_probable_state(OR_result))
    print("Business                                          Impact:",
get_most_probable_state(business_impact_result))

    overall_risk_values = OR_result.values
    overall_risk_states = OR_result.state_names['OverallRisk']

    # Розраховуємо суму всіх ймовірностей
    total_sum = sum(overall_risk_values)

    # Нормалізуємо кожне значення до інтервалу 0-1
    normalized_overall_risk = {state: prob / total_sum for state,
prob in zip(overall_risk_states, overall_risk_values)}

    # Вивід нормалізованих ймовірностей для перевірки
    print("Нормалізовані    ймовірності    для    OverallRisk:",
normalized_overall_risk)

    business_impact_values = business_impact_result.values
    business_impact_states                                          =
business_impact_result.state_names['BusinessImpact']

    # Розраховуємо суму всіх ймовірностей
    total_sum = sum(business_impact_values)

    # Нормалізуємо кожне значення до інтервалу 0-1
    normalized_business_impact = {state: prob / total_sum for
state, prob in zip(business_impact_states, business_impact_values)}

    # Вивід нормалізованих ймовірностей для перевірки
    print("Нормалізовані   ймовірності   для   BusinessImpact:",
normalized_business_impact)

    return normalized_overall_risk, normalized_business_impact
```

Опис функціоналу - Функція markov_network призначена для побудови Марковської мережі, яка використовується для оцінки ризиків інформаційної безпеки. Вона виконує комплексні операції з побудови моделі, визначення

зв'язків між змінними, створення факторів, проведення інференції та нормалізації результатів. Нижче наведено детальний опис роботи цієї функції:

Параметри функції:

- **test_scenario (dict, необов'язково):** Словник, що містить дані про наявні докази (елементи), на основі яких буде виконуватися інференція (виведення висновків). Якщо test_scenario не заданий, функція доповнить його першими станами кожної змінної.

Повертає:

- **normalized_overall_risk (dict):** Нормалізований розподіл ймовірностей для змінної OverallRisk.
- **normalized_business_impact (dict):** Нормалізований розподіл ймовірностей для змінної BusinessImpact.

Опис роботи функції:

1. **Визначення імен станів змінних:** Створюється словник state_names, який містить можливі стани для кожної змінної в мережі. Це забезпечує чітке визначення можливих значень для кожної змінної.
2. **Доповнення тестового сценарію:** Функція complete_test_scenario порівнює переданий test_scenario зі словником state_names і додає відсутні ключі зі значенням першого стану відповідної змінної. Це гарантує, що всі змінні мають визначені значення для інференції.
3. **Створення Марковської мережі:**
   - **Ініціалізація моделі:** Створюється об'єкт MarkovModel з бібліотеки pgmpy, який представляє собою Марковську мережу.
   - **Визначення змінних:** Перелічуються всі змінні, які будуть вузлами в мережі.
   - **Додавання вузлів:** Всі змінні додаються до моделі як вузли.
   - **Додавання ребер (ненаправлених зв'язків):** Визначається перелік ребер, які представляють зв'язки між змінними. Ці зв'язки відображають залежності між різними факторами ризику та їхніми впливами.
4. **Визначення кардинальності змінних:** Створюється словник cardinality, який містить кількість можливих станів для кожної змінної.

5. **Визначення кореневих змінних:** Кореневі змінні – це ті, які не залежать від інших змінних у мережі. Вони використовуються для початкового встановлення потенціалів.

6. **Створення та додавання факторів для кореневих змінних:**

- **Функція create_initial_factor:** Ця функція створює та додає фактор до моделі для початкової змінної без впливових змінних. Якщо потенціал не заданий, використовується рівномірний розподіл.
- **Визначення потенціалів:** Для кожної кореневої змінної задаються конкретні потенціали (ймовірності) або використовується рівномірний розподіл за замовчуванням.
- **Додавання факторів:** Створені фактори додаються до моделі, що визначає ймовірнісний розподіл для кореневих змінних.

7. **Створення та додавання факторів для змінних з впливовими змінними:**

- **Функція create_and_add_factor:** Ця функція створює та додає фактор до моделі для заданої цільової змінної та її впливових змінних. Вона визначає потенціали на основі логіки, де більше половини впливових змінних у стані 'High' призводить до 'High' стану цільової змінної, більше половини у стані 'Low' – до 'Low', інакше – до 'Medium'.
- **Генерація комбінацій станів:** Створюються всі можливі комбінації станів цільової змінної та її впливових змінних.
- **Визначення потенціалів:** Для кожної комбінації визначається потенціал на основі заданої логіки.
- **Додавання факторів:** Створені фактори додаються до моделі, встановлюючи залежності між змінними.

8. **Перевірка коректності моделі:** Використовується метод check_model, щоб переконатися, що мережа побудована правильно і всі фактори узгоджені. У разі виявлення помилок виводиться відповідне повідомлення.

9. **Інференція (виведення висновків):**

- **Ініціалізація механізму інференції:** Створюється об'єкт VariableElimination для виконання інференції в Марковській мережі.

- **Виконання запитів:** Виконується серія запитів для обчислення ймовірнісних розподілів для проміжних змінних, таких як UnauthorizedAccessProbability, DataInterceptionProbability тощо, з урахуванням доказів з test_scenario.
- **Визначення найбільш ймовірних станів:** Для кожного отриманого розподілу визначається стан з найвищою ймовірністю.
- **Обчислення OverallRisk та BusinessImpact:** На основі отриманих проміжних результатів виконується додатковий запит для обчислення OverallRisk, а потім – BusinessImpact.

10. **Нормалізація ймовірностей:**
    - **Для OverallRisk:** Обчислюється сума всіх ймовірностей та нормалізується кожне значення до інтервалу [0, 1], щоб сума ймовірностей дорівнювала 1.
    - **Для BusinessImpact:** Аналогічно нормалізуються ймовірності для BusinessImpact.
11. **Вивід результатів:** Виводяться постеріорні розподіли для OverallRisk та BusinessImpact, а також їхні нормалізовані значення для перевірки.

Висновок:

Функція markov_network забезпечує комплексний підхід до оцінки ризиків інформаційної безпеки за допомогою Марковської мережі. Вона дозволяє моделювати складні залежності між різними факторами ризику, проводити інференцію для визначення ймовірності загального ризику та його впливу на бізнес. Завдяки функціям для автоматичного доповнення тестового сценарію та створенню факторів, функція є гнучкою та адаптивною до різних сценаріїв оцінки ризиків.

Додаткові Особливості:

- **Моралізація:** У мережі реалізовано процес моралізації, де між батьківськими змінними додаються додаткові зв'язки, що дозволяє враховувати взаємозв'язки між впливовими змінними.

- **Гнучкість Потенціалів:** Потенціали можуть бути визначені як фіксованими значеннями, так і обчисленими на основі логіки залежностей між змінними, що дозволяє моделювати різні сценарії поведінки системи.
- **Візуалізація:** Хоча функція не включає безпосередньо візуалізацію, структура мережі та факторів може бути візуалізована за допомогою відповідних бібліотек, таких як networkx та matplotlib, для кращого розуміння взаємозв'язків між змінними.
- **Інтерактивність:** Функція дозволяє задавати власні сценарії доказів, що робить її корисною для аналізу різних ситуацій та умов безпеки.

Лістинг 3.3 – Програмна реалізація об'єднаних моделей

```
    1)Install all of the neccesarry modules
    """

    !pip install pgmpy
    !pip install networkx
    !pip install scikit-fuzzy
    !pip install pgmpy

    """1.1) Install all of the neccesarry libraries"""

    import skfuzzy as fuzz
    from skfuzzy import control as ctrl
    import matplotlib.pyplot as plt
    import networkx as nx
    from pgmpy.models import BayesianNetwork
    from pgmpy.factors.discrete import TabularCPD
    from pgmpy.inference import VariableElimination
    import numpy as np
    from pgmpy.models import MarkovModel
    from pgmpy.factors.discrete import DiscreteFactor
    from pgmpy.inference import BeliefPropagation
    from itertools import product
    import matplotlib.patches as mpatches

    def bayesian_network(test_scenario=None, seed=None):
    """
    Builds a Bayesian Network for Information Security Risk
Assessment,
    visualizes it, and performs inference based on a test scenario.

    Parameters:
    - test_scenario (dict): A dictionary representing the evidence
for inference.
```

```
    - seed (int): Seed for random number generation for
reproducibility.

    Returns:
    - posterior_overall_risk
(pgmpy.factors.discrete.DiscreteFactor): Posterior distribution of
OverallRisk.
    - posterior_business_impact
(pgmpy.factors.discrete.DiscreteFactor): Posterior distribution of
BusinessImpact.
    """

    if seed is not None:
    np.random.seed(seed)

    def generate_random_cpd(variable_card, evidence_card):
    num_columns = np.prod(evidence_card)
    cpd_values = np.random.rand(variable_card, num_columns)
    cpd_values /= cpd_values.sum(axis=0)  # Normalize columns to
sum to 1
    return cpd_values.tolist()

    # Define the structure of the Bayesian Network
    model = BayesianNetwork([
    # Dependencies for Unauthorized Access Probability
    ('PasswordStrength', 'UnauthorizedAccessProbability'),
    ('SecurityPolicies', 'UnauthorizedAccessProbability'),
    ('EncryptionKeyManagement', 'UnauthorizedAccessProbability'),

    # Dependencies for Data Interception Probability
    ('EncryptionLevel', 'DataInterceptionProbability'),
    ('EncryptionKeyManagement', 'DataInterceptionProbability'),
    ('DataSensitivity', 'DataInterceptionProbability'),

    # Dependencies for MITM Attack Probability
    ('EncryptionLevel', 'MITMAttackProbability'),
    ('EncryptionKeyManagement', 'MITMAttackProbability'),
    ('DataSensitivity', 'MITMAttackProbability'),

    # Dependencies for Attack Success Probability
    ('MalwareDetectionSystems', 'AttackSuccessProbability'),
    ('NetworkSegmentation', 'AttackSuccessProbability'),
    ('FrequencySoftwareUpdates', 'AttackSuccessProbability'),

    # Dependencies for Vulnerability Exploitation Probability
    ('FrequencySoftwareUpdates',
'VulnerabilityExploitationProbability'),
    ('MalwareDetectionSystems',
'VulnerabilityExploitationProbability'),
    ('EncryptionKeyManagement',
'VulnerabilityExploitationProbability'),
    ('ComplianceLevel', 'VulnerabilityExploitationProbability'),
    ('ThirdPartyIntegrations',
'VulnerabilityExploitationProbability'),
```

```
    # Dependencies for Physical Intrusion Probability
    ('PhysicalLocationAccessPoints',
'PhysicalIntrusionProbability'),
    ('PhysicalSecurityMeasures', 'PhysicalIntrusionProbability'),
    ('NetworkSegmentation', 'PhysicalIntrusionProbability'),

    # Dependencies for Overall Risk
    ('AttackSuccessProbability', 'OverallRisk'),
    ('DataInterceptionProbability', 'OverallRisk'),
    ('UnauthorizedAccessProbability', 'OverallRisk'),
    ('VulnerabilityExploitationProbability', 'OverallRisk'),
    ('PhysicalIntrusionProbability', 'OverallRisk'),
    ('ThirdPartyIntegrations', 'OverallRisk'),
    ('MITMAttackProbability', 'OverallRisk'),

    # Dependencies for Business Impact
    ('OverallRisk', 'BusinessImpact'),
    ('DataSensitivity', 'BusinessImpact'),
    ('BackupFrequency', 'BusinessImpact')
    ])

    # Define CPDs for each variable

    # 1. PasswordStrength
    cpd_password_strength = TabularCPD(
    variable='PasswordStrength',
    variable_card=3,
    values=[
    [0.3],  # Weak
    [0.5],  # Medium
    [0.2]   # Strong
    ],
    state_names={'PasswordStrength': ['Weak', 'Medium', 'Strong']}
    )

    # 2. SecurityPolicies
    cpd_security_policies = TabularCPD(
    variable='SecurityPolicies',
    variable_card=3,
    values=[
    [0.4],  # None
    [0.4],  # Partial
    [0.2]   # Full
    ],
    state_names={'SecurityPolicies': ['None', 'Partial', 'Full']}
    )

    # 3. EncryptionKeyManagement
    cpd_encryption_key_management = TabularCPD(
    variable='EncryptionKeyManagement',
    variable_card=3,
    values=[
    [0.5],  # Low
    [0.3],  # Medium
    [0.2]   # High
```

```
    ],
    state_names={'EncryptionKeyManagement':   ['Low',   'Medium',
'High']}
    )

    # 4. UnauthorizedAccessProbability
    cpd_unauthorized_access = TabularCPD(
    variable='UnauthorizedAccessProbability',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[3,
3, 3]),
    evidence=['PasswordStrength',             'SecurityPolicies',
'EncryptionKeyManagement'],
    evidence_card=[3, 3, 3],
    state_names={
    'UnauthorizedAccessProbability': ['Low', 'Medium', 'High'],
    'PasswordStrength': ['Weak', 'Medium', 'Strong'],
    'SecurityPolicies': ['None', 'Partial', 'Full'],
    'EncryptionKeyManagement': ['Low', 'Medium', 'High']
    }
    )

    # 5. EncryptionLevel
    cpd_encryption_level = TabularCPD(
    variable='EncryptionLevel',
    variable_card=4,
    values=[
    [0.25],  # WEP
    [0.35],  # WPA
    [0.3],   # WPA2
    [0.1]    # WPA3
    ],
    state_names={'EncryptionLevel':   ['WEP',   'WPA',   'WPA2',
'WPA3']}
    )

    # 6. DataSensitivity
    cpd_data_sensitivity = TabularCPD(
    variable='DataSensitivity',
    variable_card=3,
    values=[
    [0.4],  # Low
    [0.4],  # Medium
    [0.2]   # High
    ],
    state_names={'DataSensitivity': ['Low', 'Medium', 'High']}
    )

    # 7. DataInterceptionProbability
    cpd_data_interception = TabularCPD(
    variable='DataInterceptionProbability',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[4,
3, 3]),
```

```
    evidence=['EncryptionLevel',        'EncryptionKeyManagement',
'DataSensitivity'],
    evidence_card=[4, 3, 3],
    state_names={
    'DataInterceptionProbability': ['Low', 'Medium', 'High'],
    'EncryptionLevel': ['WEP', 'WPA', 'WPA2', 'WPA3'],
    'EncryptionKeyManagement': ['Low', 'Medium', 'High'],
    'DataSensitivity': ['Low', 'Medium', 'High']
    }
    )

    # 8. MITMAttackProbability
    cpd_mitm_attack = TabularCPD(
    variable='MITMAttackProbability',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[4,
3, 3]),
    evidence=['EncryptionLevel',        'EncryptionKeyManagement',
'DataSensitivity'],
    evidence_card=[4, 3, 3],
    state_names={
    'MITMAttackProbability': ['Low', 'Medium', 'High'],
    'EncryptionLevel': ['WEP', 'WPA', 'WPA2', 'WPA3'],
    'EncryptionKeyManagement': ['Low', 'Medium', 'High'],
    'DataSensitivity': ['Low', 'Medium', 'High']
    }
    )

    # 9. MalwareDetectionSystems
    cpd_malware_detection = TabularCPD(
    variable='MalwareDetectionSystems',
    variable_card=3,
    values=[
    [0.5],  # Low
    [0.3],  # Medium
    [0.2]   # High
    ],
    state_names={'MalwareDetectionSystems':   ['Low',   'Medium',
'High']}
    )

    # 10. NetworkSegmentation
    cpd_network_segmentation = TabularCPD(
    variable='NetworkSegmentation',
    variable_card=3,
    values=[
    [0.3],  # None
    [0.5],  # Partial
    [0.2]   # Full
    ],
    state_names={'NetworkSegmentation': ['Low', 'Medium', 'High']}
    )

    # 11. FrequencySoftwareUpdates
    cpd_frequency_software_updates = TabularCPD(
```

```
    variable='FrequencySoftwareUpdates',
    variable_card=3,
    values=[
    [0.4],  # Low
    [0.4],  # Medium
    [0.2]   # High
    ],
    state_names={'FrequencySoftwareUpdates':   ['Low',   'Medium',
'High']}
    )

    # 12. AttackSuccessProbability
    cpd_attack_success = TabularCPD(
    variable='AttackSuccessProbability',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[3,
3, 3]),
    evidence=['MalwareDetectionSystems',    'NetworkSegmentation',
'FrequencySoftwareUpdates'],
    evidence_card=[3, 3, 3],
    state_names={
    'AttackSuccessProbability': ['Low', 'Medium', 'High'],
    'MalwareDetectionSystems': ['Low', 'Medium', 'High'],
    'NetworkSegmentation': ['Low', 'Medium', 'High'],
    'FrequencySoftwareUpdates': ['Low', 'Medium', 'High']
    }
    )

    # 13. VulnerabilityExploitationProbability
    cpd_vulnerability_exploitation = TabularCPD(
    variable='VulnerabilityExploitationProbability',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[3,
3, 3, 3, 3]),
    evidence=['FrequencySoftwareUpdates',
'MalwareDetectionSystems',               'EncryptionKeyManagement',
'ComplianceLevel', 'ThirdPartyIntegrations'],
    evidence_card=[3, 3, 3, 3, 3],
    state_names={
    'VulnerabilityExploitationProbability':   ['Low',   'Medium',
'High'],
    'FrequencySoftwareUpdates': ['Low', 'Medium', 'High'],
    'MalwareDetectionSystems': ['Low', 'Medium', 'High'],
    'EncryptionKeyManagement': ['Low', 'Medium', 'High'],
    'ComplianceLevel': ['Low', 'Medium', 'High'],
    'ThirdPartyIntegrations': ['Low', 'Medium', 'High']
    }
    )

    # 14. PhysicalLocationAccessPoints
    cpd_physical_location = TabularCPD(
    variable='PhysicalLocationAccessPoints',
    variable_card=2,
    values=[
    [0.6],  # Open
```

```
    [0.4]   # Protected
    ],
    state_names={'PhysicalLocationAccessPoints':          ['Open',
'Protected']}
    )

    # 15. PhysicalSecurityMeasures
    cpd_physical_security_measures = TabularCPD(
    variable='PhysicalSecurityMeasures',
    variable_card=3,
    values=[
    [0.5],  # Low
    [0.3],  # Medium
    [0.2]   # High
    ],
    state_names={'PhysicalSecurityMeasures':   ['Low',   'Medium',
'High']}
    )

    # 16. PhysicalIntrusionProbability
    cpd_physical_intrusion = TabularCPD(
    variable='PhysicalIntrusionProbability',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[2,
3, 3]),
    evidence=['PhysicalLocationAccessPoints',
'PhysicalSecurityMeasures', 'NetworkSegmentation'],
    evidence_card=[2, 3, 3],
    state_names={
    'PhysicalIntrusionProbability': ['Low', 'Medium', 'High'],
    'PhysicalLocationAccessPoints': ['Open', 'Protected'],
    'PhysicalSecurityMeasures': ['Low', 'Medium', 'High'],
    'NetworkSegmentation': ['Low', 'Medium', 'High']
    }
    )

    # 17. OverallRisk
    cpd_overall_risk = TabularCPD(
    variable='OverallRisk',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[3,
3, 3, 3, 3, 3, 3]),
    evidence=['AttackSuccessProbability',
'DataInterceptionProbability', 'UnauthorizedAccessProbability',
    'VulnerabilityExploitationProbability',
'PhysicalIntrusionProbability',
    'ThirdPartyIntegrations','MITMAttackProbability'],
    evidence_card=[3, 3, 3, 3, 3, 3, 3],
    state_names={
    'OverallRisk': ['Low', 'Medium', 'High'],
    'AttackSuccessProbability': ['Low', 'Medium', 'High'],
    'DataInterceptionProbability': ['Low', 'Medium', 'High'],
    'UnauthorizedAccessProbability': ['Low', 'Medium', 'High'],
    'VulnerabilityExploitationProbability':   ['Low',   'Medium',
'High'],
```

```
    'PhysicalIntrusionProbability': ['Low', 'Medium', 'High'],
    'MITMAttackProbability': ['Low', 'Medium', 'High'],
    'ThirdPartyIntegrations': ['Low', 'Medium', 'High']
    }
    )

    # 18. ThirdPartyIntegrations
    cpd_third_party_integrations = TabularCPD(
    variable='ThirdPartyIntegrations',
    variable_card=3,
    values=[
    [0.5],  # Low
    [0.3],  # Medium
    [0.2]   # High
    ],
    state_names={'ThirdPartyIntegrations':    ['Low',    'Medium',
'High']}
    )

    # 19. ComplianceLevel
    cpd_compliance_level = TabularCPD(
    variable='ComplianceLevel',
    variable_card=3,
    values=[
    [0.3],  # Low
    [0.5],  # Medium
    [0.2]   # High
    ],
    state_names={'ComplianceLevel': ['Low', 'Medium', 'High']}
    )

    # 20. BackupFrequency
    cpd_backup_frequency = TabularCPD(
    variable='BackupFrequency',
    variable_card=3,
    values=[
    [0.4],  # Low
    [0.4],  # Medium
    [0.2]   # High
    ],
    state_names={'BackupFrequency': ['Low', 'Medium', 'High']}
    )

    # 21. BusinessImpact
    cpd_business_impact = TabularCPD(
    variable='BusinessImpact',
    variable_card=3,
    values=generate_random_cpd(variable_card=3,  evidence_card=[3,
3, 3]),
    evidence=['OverallRisk',                    'DataSensitivity',
'BackupFrequency'],
    evidence_card=[3, 3, 3],
    state_names={
    'BusinessImpact': ['Low', 'Medium', 'High'],
    'OverallRisk': ['Low', 'Medium', 'High'],
```

```
    'DataSensitivity': ['Low', 'Medium', 'High'],
    'BackupFrequency': ['Low', 'Medium', 'High']
    }
    )

    # Add all CPDs to the model
    model.add_cpds(
    cpd_password_strength,
    cpd_security_policies,
    cpd_encryption_key_management,
    cpd_unauthorized_access,
    cpd_encryption_level,
    cpd_data_sensitivity,
    cpd_data_interception,
    cpd_mitm_attack,
    cpd_malware_detection,
    cpd_network_segmentation,
    cpd_frequency_software_updates,
    cpd_attack_success,
    cpd_vulnerability_exploitation,
    cpd_physical_location,
    cpd_physical_security_measures,
    cpd_physical_intrusion,
    cpd_overall_risk,
    cpd_backup_frequency,
    cpd_business_impact,
    cpd_third_party_integrations,
    cpd_compliance_level
    )

    # Check the model for correctness
    try:
    model.check_model()
    print("Модель побудована коректно.")
    except ValueError as e:
    print(f"Модель містить помилки: {e}")
    return None, None

    # Define categories for nodes
    risk_factors = [
    'PasswordStrength',                         'SecurityPolicies',
'EncryptionKeyManagement',
    'EncryptionLevel',                          'DataSensitivity',
'MalwareDetectionSystems',
    'NetworkSegmentation', 'FrequencySoftwareUpdates',
    'PhysicalLocationAccessPoints', 'PhysicalSecurityMeasures',
    'ComplianceLevel', 'ThirdPartyIntegrations', 'BackupFrequency'
    ]

    risk_probabilities = [
    'UnauthorizedAccessProbability',
'DataInterceptionProbability',
    'MITMAttackProbability', 'AttackSuccessProbability',
    'VulnerabilityExploitationProbability',
'PhysicalIntrusionProbability'
```

```
]

consequences = [
'OverallRisk', 'BusinessImpact'
]

# Assign colors to nodes based on categories
color_map = {}
layer_map = {}
for node in model.nodes():
if node in risk_factors:
color_map[node] = 'skyblue'  # Blue for Risks
layer_map[node] = 1  # Top layer
elif node in risk_probabilities:
color_map[node] = 'gold'     # Yellow for Risk Probabilities
layer_map[node] = 2  # Middle layer
elif node in consequences:
color_map[node] = 'lightcoral'  # Red for Consequences
layer_map[node] = 3  # Bottom layer
else:
color_map[node] = 'grey'      # Grey for others
layer_map[node] = 2  # Default to middle layer

# Create the graph for visualization
graph_nx = nx.DiGraph()
graph_nx.add_edges_from(model.edges())

# Define positions manually to layer nodes
pos = {}
layer_nodes = {}
for node, layer in layer_map.items():
if layer not in layer_nodes:
layer_nodes[layer] = []
layer_nodes[layer].append(node)

# Assign vertical positions for each layer
y_positions = {1: 2, 2: 1, 3: 0}  # Adjust these values to
change spacing between layers

# Arrange nodes in each layer horizontally
for layer in sorted(layer_nodes.keys()):
nodes_in_layer = layer_nodes[layer]
if len(nodes_in_layer) == 1:
pos[nodes_in_layer[0]] = (0.5, y_positions[layer])
else:
x_spacing = 1.0 / (len(nodes_in_layer) + 1)
for i, node in enumerate(nodes_in_layer, start=1):
pos[node] = (i * x_spacing, y_positions[layer])

# Visualize the Bayesian Network
plt.figure(figsize=(25, 20))
nx.draw(
graph_nx, pos, with_labels=True,
node_color=[color_map[node] for node in graph_nx.nodes()],
node_size=3000, arrows=True, arrowstyle='->', arrowsize=20,
```

```
    font_size=12, font_weight='bold'
    )

    # Add legend
    legend_patches = [
    mpatches.Patch(color='skyblue', label='Ризики'),
    mpatches.Patch(color='gold', label='Ймовірності Ризиків'),
    mpatches.Patch(color='lightcoral', label='Наслідки')
    ]
    plt.legend(handles=legend_patches,     loc='upper     right',
fontsize=15)

    plt.title('Байєсівська Мережа для Оцінки Ризиків Інформаційної
Безпеки', fontsize=20)
    plt.axis('off')
    plt.show()

    # Perform inference using Variable Elimination
    infer = VariableElimination(model)

    # Define a default test scenario if none is provided
    if test_scenario is None:
    test_scenario = {
    'PasswordStrength': 'Weak',
    'SecurityPolicies': 'Full',
    'EncryptionKeyManagement': 'High',
    'EncryptionLevel': 'WPA3',
    'DataSensitivity': 'High',
    'MalwareDetectionSystems': 'High',
    'NetworkSegmentation': 'Full',
    'FrequencySoftwareUpdates': 'High',
    'PhysicalLocationAccessPoints': 'Protected',
    'PhysicalSecurityMeasures': 'High',
    'ComplianceLevel': 'High',
    'ThirdPartyIntegrations': 'Low'
    }

    # Perform inference for OverallRisk
    posterior_overall_risk = infer.query(
    variables=['OverallRisk'],
    evidence=test_scenario
    )

    print("\nOverall Risk Probability Distribution:")
    print(posterior_overall_risk)

    # Extract probabilities and states
    overall_risk_values = posterior_overall_risk.values
    overall_risk_states                                          =
posterior_overall_risk.state_names['OverallRisk']

    # Determine the most probable state of OverallRisk
    most_probable_overall_risk_index                             =
overall_risk_values.argmax()
```

```
    most_probable_overall_risk                                    =
overall_risk_states[most_probable_overall_risk_index]

    print(f"\nMost         Probable         Overall         Risk:
{most_probable_overall_risk}")

    # Perform inference for BusinessImpact based on OverallRisk
    posterior_business_impact = infer.query(
    variables=['BusinessImpact'],
    evidence={'OverallRisk': most_probable_overall_risk}
    )

    print("\nBusiness Impact Probability Distribution:")
    print(posterior_business_impact)


    overall_risk_values = posterior_overall_risk.values
    overall_risk_states                                           =
posterior_overall_risk.state_names['OverallRisk']

    # Розраховуємо суму всіх ймовірностей
    total_sum = sum(overall_risk_values)

    # Нормалізуємо кожне значення до інтервалу 0-1
    normalized_overall_risk = {state: prob / total_sum for state,
prob in zip(overall_risk_states, overall_risk_values)}

    business_impact_values = posterior_business_impact.values
    business_impact_states                                        =
posterior_business_impact.state_names['BusinessImpact']

    # Розраховуємо суму всіх ймовірностей
    total_sum = sum(business_impact_values)

    # Нормалізуємо кожне значення до інтервалу 0-1
    normalized_business_impact = {state: prob / total_sum for
state, prob in zip(business_impact_states, business_impact_values)}


    return normalized_overall_risk, normalized_business_impact

    def markov_network(test_scenario=None):

    state_names = {
    'PasswordStrength': ['Weak', 'Moderate', 'Strong'],
    'SecurityPolicies': ['None', 'Partial', 'Full'],
    'EncryptionKeyManagement': ['Low', 'Medium', 'High'],
    'UnauthorizedAccessProbability': ['Low', 'Medium', 'High'],
    'EncryptionLevel': ['WPA', 'WPA2', 'WPA3'],
    'DataSensitivity': ['Low', 'Medium', 'High'],
    'DataInterceptionProbability': ['Low', 'Medium', 'High'],
    'MITMAttackProbability': ['Low', 'Medium', 'High'],
    'MalwareDetectionSystems': ['Low', 'Medium', 'High'],
    'NetworkSegmentation': ['Low', 'Medium', 'High'],
    'FrequencySoftwareUpdates': ['Low', 'Medium', 'High'],
```

```
    'AttackSuccessProbability': ['Low', 'Medium', 'High'],
    'ThirdPartyIntegrations': ['Low', 'Medium', 'High'],
    'ComplianceLevel': ['Low', 'Medium', 'High'],
    'VulnerabilityExploitationProbability':   ['Low',   'Medium',
'High'],
    'PhysicalLocationAccessPoints':   ['Unprotected',   'Partial',
'Protected'],
    'PhysicalSecurityMeasures': ['Low', 'Medium', 'High'],
    'PhysicalIntrusionProbability': ['Low', 'Medium', 'High'],
    'OverallRisk': ['Low', 'Medium', 'High'],
    'BusinessImpact': ['Low', 'Medium', 'High'],
    'BackupFrequency': ['Rare', 'Occasional', 'Frequent']
    }


    def complete_test_scenario(test_scenario, state_names):
    #
    completed_scenario = test_scenario.copy()

    #
    for key, states in state_names.items():
    if key not in completed_scenario:
    #
    if states:
    #
    completed_scenario[key] = states[0]
    print(f" ключ '{key}' доданий зі значенням '{states[0]}'.")
    else:
    #
    completed_scenario[key] = None
    print(f" ключ '{key}' зі значенням None.")

    return completed_scenario

    test_scenario=complete_test_scenario(test_scenario,
state_names)

    # Створення Марковської мережі
    model = MarkovModel()

    # Перелік змінних
    variables = [
    'PasswordStrength',
    'SecurityPolicies',
    'EncryptionKeyManagement',
    'UnauthorizedAccessProbability',
    'EncryptionLevel',
    'DataSensitivity',
    'DataInterceptionProbability',
    'MITMAttackProbability',
    'MalwareDetectionSystems',
    'NetworkSegmentation',
    'FrequencySoftwareUpdates',
    'AttackSuccessProbability',
    'ThirdPartyIntegrations',
```

```
    'ComplianceLevel',
    'VulnerabilityExploitationProbability',
    'PhysicalLocationAccessPoints',
    'PhysicalSecurityMeasures',
    'PhysicalIntrusionProbability',
    'OverallRisk',
    'BusinessImpact',
    'BackupFrequency'
    ]

    # Додавання вузлів (змінних)
    model.add_nodes_from(variables)

    # Додавання ребер (ненаправлених зв'язків)
    edges = [
    # Ненаправлені ребра з Байєсівської мережі
    ('PasswordStrength', 'UnauthorizedAccessProbability'),
    ('SecurityPolicies', 'UnauthorizedAccessProbability'),
    ('EncryptionKeyManagement', 'UnauthorizedAccessProbability'),

    # Ребра між батьками (моралізація)
    ('PasswordStrength', 'SecurityPolicies'),
    ('PasswordStrength', 'EncryptionKeyManagement'),
    ('SecurityPolicies', 'EncryptionKeyManagement'),

    # Залежності для DataInterceptionProbability
    ('DataSensitivity', 'DataInterceptionProbability'),
    ('EncryptionLevel', 'DataInterceptionProbability'),
    ('EncryptionKeyManagement', 'DataInterceptionProbability'),

    # Залежності для MITMAttackProbability
    ('DataSensitivity', 'MITMAttackProbability'),
    ('EncryptionLevel', 'MITMAttackProbability'),
    ('EncryptionKeyManagement', 'MITMAttackProbability'),

    # Додаткові зв'язки між змінними
    ('DataSensitivity', 'EncryptionLevel'),
    ('DataSensitivity', 'EncryptionKeyManagement'),
    ('EncryptionLevel', 'EncryptionKeyManagement'),

    # Залежності для AttackSuccessProbability
    ('MalwareDetectionSystems', 'AttackSuccessProbability'),
    ('NetworkSegmentation', 'AttackSuccessProbability'),
    ('FrequencySoftwareUpdates', 'AttackSuccessProbability'),

    # Ребра між батьками для AttackSuccessProbability
    ('MalwareDetectionSystems', 'NetworkSegmentation'),
    ('MalwareDetectionSystems', 'FrequencySoftwareUpdates'),
    ('NetworkSegmentation', 'FrequencySoftwareUpdates'),

    # Залежності для VulnerabilityExploitationProbability
    ('MalwareDetectionSystems',
'VulnerabilityExploitationProbability'),
    ('EncryptionKeyManagement',
'VulnerabilityExploitationProbability'),
```

```
    ('FrequencySoftwareUpdates',
'VulnerabilityExploitationProbability'),
    ('ThirdPartyIntegrations',
'VulnerabilityExploitationProbability'),
    ('ComplianceLevel', 'VulnerabilityExploitationProbability'),

    # Додаткові зв'язки для VulnerabilityExploitationProbability
    ('MalwareDetectionSystems', 'EncryptionKeyManagement'),
    ('MalwareDetectionSystems', 'ThirdPartyIntegrations'),
    ('MalwareDetectionSystems', 'ComplianceLevel'),
    ('EncryptionKeyManagement', 'ThirdPartyIntegrations'),
    ('EncryptionKeyManagement', 'ComplianceLevel'),
    ('FrequencySoftwareUpdates', 'ThirdPartyIntegrations'),
    ('FrequencySoftwareUpdates', 'ComplianceLevel'),
    ('ThirdPartyIntegrations', 'ComplianceLevel'),

    # Залежності для PhysicalIntrusionProbability
    ('NetworkSegmentation', 'PhysicalIntrusionProbability'),
    ('PhysicalLocationAccessPoints',
'PhysicalIntrusionProbability'),
    ('PhysicalSecurityMeasures', 'PhysicalIntrusionProbability'),

    # Ребра між батьками для PhysicalIntrusionProbability
    ('NetworkSegmentation', 'PhysicalLocationAccessPoints'),
    ('NetworkSegmentation', 'PhysicalSecurityMeasures'),
    ('PhysicalSecurityMeasures', 'PhysicalLocationAccessPoints'),

    # Залежності для OverallRisk
    ('DataInterceptionProbability', 'OverallRisk'),
    ('UnauthorizedAccessProbability', 'OverallRisk'),
    ('MITMAttackProbability', 'OverallRisk'),
    ('AttackSuccessProbability', 'OverallRisk'),
    ('VulnerabilityExploitationProbability', 'OverallRisk'),
    ('PhysicalIntrusionProbability', 'OverallRisk'),

    # Додаткові зв'язки між змінними, що впливають на OverallRisk
    ('DataInterceptionProbability',
'UnauthorizedAccessProbability'),
    ('DataInterceptionProbability', 'MITMAttackProbability'),
    ('DataInterceptionProbability', 'AttackSuccessProbability'),
    ('DataInterceptionProbability',
'VulnerabilityExploitationProbability'),
    ('DataInterceptionProbability',
'PhysicalIntrusionProbability'),
    ('UnauthorizedAccessProbability', 'MITMAttackProbability'),
    ('UnauthorizedAccessProbability', 'AttackSuccessProbability'),
    ('UnauthorizedAccessProbability',
'VulnerabilityExploitationProbability'),
    ('UnauthorizedAccessProbability',
'PhysicalIntrusionProbability'),
    ('MITMAttackProbability', 'AttackSuccessProbability'),
    ('MITMAttackProbability',
'VulnerabilityExploitationProbability'),
    ('MITMAttackProbability', 'PhysicalIntrusionProbability'),
```

```
    ('AttackSuccessProbability',
'VulnerabilityExploitationProbability'),
    ('AttackSuccessProbability', 'PhysicalIntrusionProbability'),
    ('VulnerabilityExploitationProbability',
'PhysicalIntrusionProbability'),

    # Залежності для BusinessImpact
    ('OverallRisk','BusinessImpact'),
    ('DataSensitivity','BusinessImpact'),
    ('BackupFrequency','BusinessImpact'),

    # Додаткові зв'язки для BusinessImpact
    ('OverallRisk','DataSensitivity'),
    ('OverallRisk','BackupFrequency'),
    ('BackupFrequency','DataSensitivity'),
    ]

    model.add_edges_from(edges)

    # Визначення кардинальності для кожної змінної
    cardinality = {var: len(state_names[var]) for var in variables}

    # Визначення кореневих (root) змінних (без впливових змінних)
    non_root_variables = [
    'UnauthorizedAccessProbability',
    'DataInterceptionProbability',
    'MITMAttackProbability',
    'AttackSuccessProbability',
    'VulnerabilityExploitationProbability',
    'PhysicalIntrusionProbability',
    'OverallRisk',
    'BusinessImpact'
    ]

    root_variables = [var for var in variables if var not in
non_root_variables]

    # Функція для створення та додавання факторів для початкових
змінних
    def   create_initial_factor(model,   variable,   state_names,
cardinality, potential_values=None):
    """
    Створює та додає фактор до моделі для початкової змінної (без
впливових змінних).

    :param model: Марковська модель (MarkovModel).
    :param variable: Назва змінної (str).
    :param state_names: Словник зі станами змінних (dict).
    :param cardinality: Словник з кардинальністю змінних (dict).
    :param potential_values: Список або масив ймовірностей для
станів змінної (list або np.array).
    Якщо None, використовується рівномірний розподіл.
    """
    variables = [variable]
    card = [cardinality[variable]]
```

```
    if potential_values is None:
    # Рівномірний розподіл
    potential_values = np.ones(cardinality[variable])
    # Не нормалізуємо потенціали в Марковських мережах
    else:
    potential_values = np.array(potential_values, dtype=float)
    # Перевірка сумарної ймовірності (не обов'язково для
Марковських мереж)
    # if not np.isclose(np.sum(potential_values), 1.0):
    #     potential_values /= np.sum(potential_values)

    # Створення фактора
    factor = DiscreteFactor(
    variables,
    card,
    potential_values,
    state_names={variable: state_names[variable]}
    )

    # Додавання фактора до моделі
    model.add_factors(factor)
    print(f"Фактор для {variables} додано.\n")

    # Створення факторів для кореневих змінних
    for var in root_variables:
    potential = None
    if var == 'PasswordStrength':
    potential = [0.3, 0.5, 0.2]  # Weak, Moderate, Strong
    elif var == 'SecurityPolicies':
    potential = [0.2, 0.6, 0.2]  # None, Partial, Full
    elif var == 'EncryptionKeyManagement':
    potential = [0.4, 0.5, 0.1]  # Low, Medium, High
    elif var == 'EncryptionLevel':
    potential = [0.3, 0.6, 0.1]  # WPA, WPA2, WPA3
    elif var == 'DataSensitivity':
    potential = [0.5, 0.3, 0.2]  # Low, Medium, High
    elif var in ['MalwareDetectionSystems', 'NetworkSegmentation',
'FrequencySoftwareUpdates',
    'ComplianceLevel',                  'ThirdPartyIntegrations',
'PhysicalLocationAccessPoints',
    'PhysicalSecurityMeasures', 'BackupFrequency']:
    potential = [0.5, 0.3, 0.2]  # Low, Medium, High або відповідно
    else:
    potential = None  # Для додаткової безпеки

    create_initial_factor(
    model=model,
    variable=var,
    state_names=state_names,
    cardinality=cardinality,
    potential_values=potential
    )
```

```
    # Функція для створення та додавання факторів для змінних з впливовими змінними
    def create_and_add_factor(model, target_var, influencer_vars, state_names, cardinality):
    """
    Створює та додає фактор до моделі для заданої цільової змінної та її впливових змінних.

    :param model: Марковська модель
    :param target_var: Цільова змінна (str)
    :param influencer_vars: Список впливових змінних (list of str)
    :param state_names: Словник зі станами змінних (dict)
    :param cardinality: Словник з кардинальністю змінних (dict)
    """
    variables = [target_var] + influencer_vars
    card = [cardinality[var] for var in variables]

    for var1 in variables:
    for var2 in variables:
    if var1 != var2 and (var1, var2) not in model.edges and (var2, var1) not in model.edges:
    model.add_edge(var1, var2)
    print(f"Додано ребро: ({var1}, {var2})")


    # Генерація всіх можливих комбінацій станів
    combinations = list(product(*[range(c) for c in card]))

    # Ініціалізація потенціалу
    data_for_potential = np.zeros(len(combinations))

    for idx, combo in enumerate(combinations):
    # combo[0] - target_var state
    # combo[1:] - influencing_vars states
    target_state = combo[0]
    influencing_states = combo[1:]

    # Логіка потенціалу:
    # Якщо більше половини впливових змінних в стані 'High', target_var в стані 'High'
    # Якщо більше половини впливових змінних в стані 'Low', target_var в стані 'Low'
    # Інакше target_var в стані 'Medium'

    high_count = sum(state == (cardinality[var]-1) for var, state in zip(influencer_vars, influencing_states))
    low_count = sum(state == 0 for state in influencing_states)
    total = len(influencer_vars)

    if high_count > total / 2 and target_state == (cardinality[target_var]-1):
    potential = 10.0  # Високий потенціал для 'High'
    elif low_count > total / 2 and target_state == 0:
    potential = 10.0  # Високий потенціал для 'Low'
    elif target_state == 1:
```

```
    potential = 5.0   # Середній потенціал для 'Medium'
    else:
    potential = 1.0   # Низький потенціал для інших випадків

    data_for_potential[idx] = potential

    # Не нормалізуємо потенціали в Марковських мережах
    # data_for_potential /= np.sum(data_for_potential)

    # Створення фактора
    factor = DiscreteFactor(
    variables,
    card,
    data_for_potential,
    state_names={var: state_names[var] for var in variables}
    )

    # Додавання фактора до моделі
    model.add_factors(factor)
    print(f"Фактор для {variables} додано.\n")

    # Створення факторів для змінних з впливовими змінними
    create_and_add_factor(
    model=model,
    target_var='UnauthorizedAccessProbability',
    influencer_vars=['PasswordStrength',       'SecurityPolicies',
'EncryptionKeyManagement'],
    state_names=state_names,
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='DataInterceptionProbability',
    influencer_vars=['EncryptionLevel', 'EncryptionKeyManagement',
'DataSensitivity'],
    state_names=state_names,
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='MITMAttackProbability',
    influencer_vars=['EncryptionLevel', 'EncryptionKeyManagement',
'DataSensitivity'],
    state_names=state_names,
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='AttackSuccessProbability',
    influencer_vars=['MalwareDetectionSystems',
'NetworkSegmentation', 'FrequencySoftwareUpdates'],
    state_names=state_names,
```

```
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='VulnerabilityExploitationProbability',
    influencer_vars=['FrequencySoftwareUpdates',
'MalwareDetectionSystems',              'EncryptionKeyManagement',
'ThirdPartyIntegrations', 'ComplianceLevel'],
    state_names=state_names,
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='PhysicalIntrusionProbability',
    influencer_vars=['NetworkSegmentation','PhysicalLocationAcces
sPoints','PhysicalSecurityMeasures'],
    state_names=state_names,
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='OverallRisk',
    influencer_vars=[
    'DataInterceptionProbability',
    'UnauthorizedAccessProbability',
    'MITMAttackProbability',
    'AttackSuccessProbability',
    'VulnerabilityExploitationProbability',
    'PhysicalIntrusionProbability'
    ],
    state_names=state_names,
    cardinality=cardinality
    )

    create_and_add_factor(
    model=model,
    target_var='BusinessImpact',
    influencer_vars=['OverallRisk',            'DataSensitivity',
'BackupFrequency'],
    state_names=state_names,
    cardinality=cardinality
    )

    # Перевірка, чи всі фактори створені для кореневих змінних
    # Оскільки кореневі змінні вже мають фактори, немає потреби
додавати додаткові фактори
    # Це уникає дублювання факторів для змінних з впливовими
змінними

    # Додатково, можна переконатися, що немає факторів для
non_root_variables
    for var in non_root_variables:
```

```
    if model.get_factors(var):
    print(f"Попередження: Фактор для {var} вже існує як частина
іншого фактора.")

    # Перевірка моделі
    if model.check_model():
    print("Модель коректна.")
    else:
    print("Модель некоректна. Перевірте фактори та їхні
потенціали.")

    # Створення об'єкта інференції



    from pgmpy.inference import VariableElimination

    # Створення об'єкта інференції
    infer = VariableElimination(model)

    #infer = BeliefPropagation(model)

    # Встановлення доказів (Evidence)
    evidence = {

    }

    # Виконання запиту
    #result                                                    =
infer.query(variables=['DataSensitivity','OverallRisk','BackupFreq
uency','BusinessImpact'], show_progress=True)
    #map_result                                                =
infer.map_query(variables=['DataSensitivity','OverallRisk','Backup
Frequency','BusinessImpact'], evidence=evidence)

    # Виведення результатів
    #print("Найбільш ймовірна комбінація станів:")
    #for var, state in map_result.items():
    #    print(f"{var}: {state}")

    UAP_result                                                =
infer.query(variables=['UnauthorizedAccessProbability'],
evidence={'PasswordStrength':    test_scenario['PasswordStrength'],
'SecurityPolicies':              test_scenario['SecurityPolicies'],
'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement']}, show_progress=True)
    DIP_result                                                =
infer.query(variables=['DataInterceptionProbability'],
evidence={'EncryptionLevel':      test_scenario['EncryptionLevel'],
'DataSensitivity':                test_scenario['DataSensitivity'],
'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement']}, show_progress=True)
    MitmAP_result                                             =
infer.query(variables=['MITMAttackProbability'],
evidence={'PasswordStrength':    test_scenario['PasswordStrength'],
```

```
'SecurityPolicies':              test_scenario['SecurityPolicies'],
'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement']}, show_progress=True)
     ASP_result                                                    =
infer.query(variables=['AttackSuccessProbability'],
evidence={'MalwareDetectionSystems':
test_scenario['MalwareDetectionSystems'],    'NetworkSegmentation':
test_scenario['NetworkSegmentation'],   'FrequencySoftwareUpdates':
test_scenario['FrequencySoftwareUpdates']}, show_progress=True)
     VEP_result                                                    =
infer.query(variables=['VulnerabilityExploitationProbability'],
evidence={'EncryptionKeyManagement':
test_scenario['EncryptionKeyManagement'],
'MalwareDetectionSystems':
test_scenario['MalwareDetectionSystems'],
'FrequencySoftwareUpdates':
test_scenario['FrequencySoftwareUpdates'],       'ComplianceLevel':
test_scenario['ComplianceLevel'],'ThirdPartyIntegrations':
test_scenario['ThirdPartyIntegrations']}, show_progress=True)
     PIP_result                                                    =
infer.query(variables=['PhysicalIntrusionProbability'],
evidence={'NetworkSegmentation':
test_scenario['NetworkSegmentation'],
'PhysicalLocationAccessPoints':
test_scenario['PhysicalLocationAccessPoints'],
'PhysicalSecurityMeasures':
test_scenario['PhysicalSecurityMeasures']}, show_progress=True)


     #M2_result  =  infer.query(variables=['M2'],  evidence={'B4':
'value4', 'B5': 'value5', 'B6': 'value6'}, show_progress=True)
     # Повторіть для кожної середньої змінної M3, M4, ..., M6

     # Зберігаємо результати часткових симуляцій середніх змінних в
словнику для подальшого використання
     def get_most_probable_state(result):
     max_index  =  result.values.argmax()     #  Індекс  стану  з
максимальною ймовірністю
     return  result.state_names[result.variables[0]][max_index]   #
Назва стану

     partial_results = {
     'UAP': get_most_probable_state(UAP_result),
     'DIP': get_most_probable_state(DIP_result),
     'MitmAP': get_most_probable_state(MitmAP_result),
     'ASP': get_most_probable_state(ASP_result),
     'VEP': get_most_probable_state(VEP_result),
     'PIP': get_most_probable_state(PIP_result),
     }
     #print(partial_results)

     OR_result       =       infer.query(variables=['OverallRisk'],
evidence={'UnauthorizedAccessProbability':  partial_results['UAP'],
'DataInterceptionProbability': partial_results['DIP'],
     'MITMAttackProbability': partial_results['MitmAP'] ,
```

```
    'AttackSuccessProbability': partial_results['ASP'],
    'VulnerabilityExploitationProbability':
partial_results['VEP'],
    'PhysicalIntrusionProbability': partial_results['PIP']
    }, show_progress=True)

    #print(OR_result)
    business_impact_result = infer.query(
    variables=['BusinessImpact'],
    evidence={'OverallRisk':   get_most_probable_state(OR_result),
'DataSensitivity':                test_scenario['DataSensitivity'],
'BackupFrequency': test_scenario['BackupFrequency']},
    show_progress=True
    )

    print("Overall Risk:", OR_result)
    print("Business Impact:", business_impact_result)
    print("Overall Risk:", get_most_probable_state(OR_result))
    print("Business                                        Impact:",
get_most_probable_state(business_impact_result))

    overall_risk_values = OR_result.values
    overall_risk_states = OR_result.state_names['OverallRisk']

    # Розраховуємо суму всіх ймовірностей
    total_sum = sum(overall_risk_values)

    # Нормалізуємо кожне значення до інтервалу 0-1
    normalized_overall_risk = {state: prob / total_sum for state,
prob in zip(overall_risk_states, overall_risk_values)}

    # Вивід нормалізованих ймовірностей для перевірки
    print("Нормалізовані    ймовірності    для    OverallRisk:",
normalized_overall_risk)

    business_impact_values = business_impact_result.values
    business_impact_states                                        =
business_impact_result.state_names['BusinessImpact']

    # Розраховуємо суму всіх ймовірностей
    total_sum = sum(business_impact_values)

    # Нормалізуємо кожне значення до інтервалу 0-1
    normalized_business_impact  =  {state:  prob  /  total_sum  for
state, prob in zip(business_impact_states, business_impact_values)}

    # Вивід нормалізованих ймовірностей для перевірки
    print("Нормалізовані   ймовірності   для   BusinessImpact:",
normalized_business_impact)

    return normalized_overall_risk, normalized_business_impact

    test_scenario = {
    'PasswordStrength': 'Weak',
    'SecurityPolicies': 'Full',
```

```
    'EncryptionKeyManagement': 'High',
    'EncryptionLevel': 'WPA3',
    'DataSensitivity': 'High',
    'MalwareDetectionSystems': 'High',
    'NetworkSegmentation': 'High',
    'FrequencySoftwareUpdates': 'High',
    'PhysicalLocationAccessPoints': 'Protected',
    'PhysicalSecurityMeasures': 'High',
    'ComplianceLevel': 'High',
    'ThirdPartyIntegrations': 'Low'
    }

    markov_res_risk,markov_res_business=markov_network(test_scena
rio=test_scenario)
    bayes_res_risk,bayes_res_business=bayesian_network(test_scena
rio=test_scenario)

    print(markov_res_risk)
    print(markov_res_business)
    print(bayes_res_risk)
    print(bayes_res_business)

    def simple_averaging(bayes_res, markov_res):
    unified_result = {}

    # Compute the simple average for each category
    for risk_level in bayes_res:
    unified_result[risk_level]    =    (bayes_res[risk_level]    +
markov_res[risk_level]) / 2

    # Display the unified result
    return unified_result



    def weighted_averaging(bayes_res, markov_res):
    weight_bayesian = 0.6  # 60% weight to Bayesian
    weight_markov = 0.4    # 40% weight to Markov

    # Ensure that weights sum to 1
    total_weight = weight_bayesian + weight_markov
    weight_bayesian /= total_weight
    weight_markov /= total_weight

    # Initialize the unified result dictionary
    unified_weighted = {}

    # Compute the weighted average for each category
    for risk_level in bayes_res:
    unified_weighted[risk_level] = (
    bayes_res[risk_level] * weight_bayesian +
    markov_res[risk_level] * weight_markov
    )

    # Display the unified weighted result
```

```
    return unified_weighted

    def multiplicative_combination(bayes_res, markov_res):
    unified_multiplicative = {}

    # Multiply the probabilities for each category
    for risk_level in bayes_res:
    unified_multiplicative[risk_level] = bayes_res[risk_level] *
markov_res[risk_level]

    # Calculate the total sum for normalization
    total = sum(unified_multiplicative.values())

    # Normalize the multiplied probabilities
    for risk_level in unified_multiplicative:
    unified_multiplicative[risk_level] /= total

    # Display the unified multiplicative result
    return unified_multiplicative

    print("Using simple averaging method:")
    print("For risk")
    print(simple_averaging(bayes_res_risk, markov_res_risk))
    print("For business impact")
    print(simple_averaging(bayes_res_business,
markov_res_business))
    print("-----------------------------------------------------")
    print("Using weighted averaging method:")
    print("For risk")
    print(weighted_averaging(bayes_res_risk, markov_res_risk))
    print("For business impact")
    print(weighted_averaging(bayes_res_business,
markov_res_business))
    print("-----------------------------------------------------")
    print("Using multiplicative_combination method:")
    print("For risk")
    print(multiplicative_combination(bayes_res_risk,
markov_res_risk))
    print("For business impact")
    print(multiplicative_combination(bayes_res_business,
markov_res_business))
    print("-----------------------------------------------------")
```

Результати роботи об'єднаного коду:

```
Using simple averaging method:
For risk
{'Low': 0.19602673583035668, 'Medium': 0.29206846666558783, 'High': 0.5119047975040555}
For business impact
{'Low': 0.23046458045560533, 'Medium': 0.32407082358918765, 'High': 0.44546459595520704}
-----------------------------------------------------
Using weighted averaging method:
For risk
{'Low': 0.22202211895154933, 'Medium': 0.30628109503608836, 'High': 0.4716967860123624}
For business impact
{'Low': 0.2640574965467264, 'Medium': 0.32638498830702517, 'High': 0.4095575151462485}
-----------------------------------------------------
Using multiplicative_combination method:
For risk
{'Low': 0.06657830158373856, 'Medium': 0.24814492564773932, 'High': 0.6852767727685222}
For business impact
{'Low': 0.08412903452189435, 'Medium': 0.35435668988028257, 'High': 0.5615142755978232}
-----------------------------------------------------
```

Рисунок 3.1 Результат роботи об'єднаного коду

СПИСОК ДЖЕРЕЛ

1. **Probabilistic Graphical Models: Principles and Techniques**. Daphne Koller, Nir Friedman. URL: http://mcb111.org/w06/KollerFriedman.pdf (дата звернення: 01.11.2024).
2. **Bayesian Networks and Decision Graphs**. Finn V. Jensen, Thomas D. Nielsen. URL:http://ccomputo.unsaac.edu.pe/wiki/downloads/Bayesian%20Networks%20and%20Decision%20Graphs%202ed_Finn%20V.%20Jensen%20(Springer%202007%20457s).pdf (дата звернення: 02.11.2024).
3. **Fuzzy Logic with Engineering Applications**. Timothy J. Ross. URL: https://home.iitk.ac.in/~avrs/ManyValuedLogic/FuzzyLogicforEngineers.pdf (дата звернення: 03.11.2024).
4. **Markov Random Fields for Vision and Image Processing**. P. W. Sauer. URL: https://direct.mit.edu/books/edited-volume/2128/Markov-Random-Fields-for-Vision-and-Image (дата звернення: 04.11.2024).
5. **pgmpy: A Python library for Probabilistic Graphical Models**. pgmpy Documentation.URL: https://pgmpy.org/ (дата звернення: 05.11.2024).
6. **Fuzzy Logic Introduction**. Surya Priy, Abhishek Rajput. URL: https://www.geeksforgeeks.org/fuzzy-logic-introduction/ (дата звернення: 06.11.2024).
7. **Python for Probability, Statistics, and Machine Learning**. José Unpingco. URL: http://54.243.252.9/engr-1330-webroot/3-Readings/Python-for-Probability-Statistics-And-Machine-Learning.pdf (дата звернення: 07.11.2024).
8. **Markov Random Fields and Graphical Models in Python**. Journal Article. URL: https://www.cs.umd.edu/~gihan/resources/em509/project/markov-random-fields.pdf (дата звернення: 08.11.2024).
9. **Integrating Bayesian Networks with Markov Random Fields**. Research Paper. URL: https://home.ttic.edu/~dmcallester/ttic101-06/lectures/graphmodels/graphmodels.pdf (дата звернення: 09.11.2024).
10. **Machine Learning with Bayesian Networks in Python**. Zulaikha Lateef. URL: https://www.edureka.co/blog/bayesian-networks/ (дата звернення: 11.11.2024).
11. **Graphical Models in Python: An Introduction**. Almero Gouws. URL: https://scholar.sun.ac.za/server/api/core/bitstreams/40429eea-73ee-4055-8bef-5d989aaff8fe/content (дата звернення: 12.11.2024).
12. **Deep Learning with Markov Random Fields and Bayesian Networks**. Zhenxing Wang, L. Chan . URL: https://www.semanticscholar.org/paper/Learning-bayesian-networks-from-Markov-random-An-Wang-Chan/aa18b106ab3d65c4952bd8ea2833fba86492597d (дата звернення: 14.11.2024).
13. **Applying Fuzzy Logic to Risk Assessment and Decision-Making**. Kailan Shang, Zakir Hossen . URL: https://www.casact.org/sites/default/files/2021-02/research_applying-fuzzy-logic.pdf (дата звернення: 14.11.2024).
14. **Fuzzy decision tree of risk assessment generated from risk response**. Hafeth Ibrahim Naji.URL: https://www.researchgate.net/publication/326326690_Fuzzy_decision_tree_of_risks_assessment_generated_from_risk_response (дата звернення: 14.11.2024).

15. **An Introduction to Markov Modeling: Concepts and Uses**. Mark A. Boyd.URL:https://ntrs.nasa.gov/api/citations/20020050518/downloads/20020050518.pdf (дата звернення: 14.11.2024).
16. **Risk Assessment: An Overview .** Blog. URL: https://legal.thomsonreuters.com/blog/what-is-a-risk-assessment/ (дата звернення: 14.11.2024).
17. **Security Risk Assessment.** Blog. URL: https://www.blackduck.com/glossary/what-is-security-risk-assessment.html#:~:text=A%20security%20risk%20assessment%20identifies,holistically%E2%80%94from%20an%20attacker's%20perspective. (дата звернення: 14.11.2024).
18. **How to perform a successfull IT Risk Assessment.** Kayne McGladrey. URL: https://hyperproof.io/resource/it-risk-assessment/ (дата звернення: 16.11.2024).
19. **Mathematical Modeling and Statistical Methods for Risk Management.** Henrik Hult, Filip Lindskog. URL: https://people.kth.se/~lindskog/papers/RMlecturenotes07B.pdf (дата звернення: 17.11.2024).
20. **A new mathematical model for analytical risk assessment and prediction in IT systems.** Imed el Fray, Miroslaw Kurkowski, Jerzy Pejas. URL: https://www.researchgate.net/publication/266675218_A_new_mathematical_model_for_analytical_risk_assessment_and_prediction_in_IT_systems (дата звернення: 19.11.2024).